TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

HENRI PURANEN

KETTERÄT OHJELMISTOKEHITYSMENETELMÄT USEAN JÄRJESTELMÄTOIMITTAJAN IT-HANKKEESSA

Diplomityö

Tarkastaja: tutkija Jukka Huhtamäki
Tarkastaja ja aihe hyväksytty
Tieto- ja sähkötekniikan tiedekuntaneuvoston kokouksessa 4. marraskuuta 2015

TIIVISTELMÄ

HENRI PURANEN: Ketterät ohjelmistokehitysmenetelmät usean järjestelmätoimittajan IT-hankkeessa
Tampereen teknillinen yliopisto
Diplomityö, 55 sivua
Joulukuu 2015
Tietotekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Hypermedia
Tarkastaja: tutkija Jukka Huhtamäki

Avainsanat: ketterät ohjelmistokehitysmenetelmät, Scrum, Agile, Lean, Kanban, LeSS, Large-Scale Scrum, tapaustutkimus

Ketterät ohjelmistokehitysmenetelmät kasvattavat jatkuvasti suosiotaan IT-alan monipuolisten projektien hallinnassa. Ketterät menetelmät mahdollistavat tehokkaan ohjelmistokehityksen vaiheistamalla toteutustyön, keskittymällä olennaisiin asioihin, korostamalla tiimin ja yhteistyön merkitystä, sekä vastaamalla muuttuviin liiketoimintavaatimuksiin. Menetelmien käyttämisestä osana suuria tietojärjestelmähankkeita on hyvin vähän tutkimustuloksia saatavilla.

Tässä diplomityössä suoritetaan kattava, havainnointiin perustuva tapaustutkimus suuresta tietojärjestelmän uudistushankkeesta, jossa on mukana useita järjestelmätoimittajia, toteuttamassa kokonaisuuden yksittäisiä tietojärjestelmiä. Tutkimuksessa keskitytään selvittämään miten ketterät ohjelmistokehitysmenetelmät soveltuvat hankkeeseen, jossa on useita osapuolia, joiden välillä vaaditaan tiivistä yhteistyötä onnistuneen lopputuloksen saavuttamiseksi. Tutkimuksessa kuvataan tarkasti millä tavalla ketteriä menetelmiä käytetään hankkeessa käytännössä.

Tutkimustuloksina saavutetaan selkeitä havaintoja ketterien menetelmien soveltuvuudesta usean järjestelmätoimittajan hankkeeseen. Tuloksista nähdään erityiset haasteet, joita usean osapuolen väliseen projektinhallintaan liittyy, mutta havaitaan myös keinoja näiden haasteiden hallintaan. Työn tuloksista johdetaan jatkotutkimuskohteet ja esitetään tietojärjestelmähankkeen asiakkaalle konkreettisia toimenpiteitä siitä, millä tavalla hanketta voitaisiin kehittää projektinhallinnan näkökulmasta.

ABSTRACT

HENRI PURANEN: Usage of Agile software development methods in multi-supplier IT projects
Tampere University of Technology
Master of Science Thesis, 55 pages
December 2015
Master's Degree Programme in Information Technology
Major: Hypermedia
Examiner: Researcher Jukka Huhtamäki

Keywords: Agile software development methods, Scrum, Lean, Kanban, LeSS, Large-Scale Scrum, case study

Agile software development methods are becoming very popular ways to organize the versatile projects of software development industries. Agile methods provide various ways to enhance the efficiency of software development. These include: iterative and incremental development of features, focusing on the things that matter, emphasizing the importance of a team and co-operation and the ability to respond to change of requirements. Using the Agile methods as a part of large-scale projects is a fairly new aspect in terms of scientific research.

This thesis performs a comprehensive, descriptive case study of large-scale information system renewal project, which consists of several system suppliers. The case study aims to research how the Agile software development methods apply on multi-supplier IT-project. The project requires deep co-operation between the suppliers and their separate information systems, which are required to work together. The case study describes how the Agile methods are used in practice in the project.

The conclusions of the research conduct clear results of how these methods apply on the project. The thesis describes the challenges regarding the usage of the Agile methods and expresses how these challenges are manageable. The conclusions describe possible additional research topics. The thesis also represent real actions that should provide ways to improve the overall efficiency of the project in terms of overall project management.

ALKUSANAT

Diplomityön tekeminen on ollut minulle melko haastava prosessi, sillä edellisestä kurssisuorituksesta on kohdallani kulunut jo pari vuotta. Diplomityön aihe ehti vaihtua tänä aikana kertaalleen, mutta nyt työn valmistuttua voin olla lopputulokseen erittäin tyytyväinen. Tämän työn aihe on kypsynyt mielessäni vajaan kahden vuoden ajan työskennellessäni nykyisessä työpaikassani. Tiesin jo pitkään, että diplomityön aihe koskettaa lopulta nykyistä työnkuvaani ja projektia, jossa olen ollut mukana. Työn aihe muodostui lopulta hyvin suoraviivaisesti sen tukemana, mitä olen nykyisessä työtehtävässäni tehnyt. Toimenkuvani tässä IT-hankkeessa on tukenut tutkimuksen suorittamista ja havainnointi todella hyvin.

Haluan kiittää tutkija Jukka Huhtamäkeä, joka on ollut mukana diplomityöprosessissani alusta alkaen, myös ensimmäisen aihe-ehdotukseni kohdalla vuoden 2013 alussa. Tähän diplomityöhön olen saanut Jukalta tärkeitä näkökulmia tutkimuksen kuvaamiseen liittyen sekä hyviä vinkkejä tulosten ilmaisemiseen. Lisäksi haluan kiittää rakasta tyttöystävääni Hennaa tukemisesta vaikeina aikoina ja kannustamisesta työn tekemiseen.

Tampereella, 25.11.2015

Henri Puranen

SISÄLLYSLUETTELO

1. JOHDANTO

Kansainvälisten yritysten on jatkuvasti tehostettava toimintaansa globalisaation tuomaa muutosta varten. Yrityksien on uudistuttava ja kehityttävä, jotta ne pystyvät vastaamaan globalisaation asettamiin haasteisiin. (Enemmän, mutta parempaa globalisaatiota 2010) Eräs varteenotettava keino yritysten toiminnan tehostamiseen on monimuotoisten ja tehokkaiden tietojärjestelmien kehittäminen. Vanhojen ja kankeiden tietojärjestelmien korvaaminen nykyaikaisilla, tehokkailla ratkaisuilla voi tarjota yrityksille merkittävää kilpailuetua. Menestyjiä yhä kansainvälisemmillä markkinoilla ovat yritykset, jotka onnistuvat uudistumaan muita paremmin.

Ohjelmistotuotannossa on 2000 luvun alkuvuosista alkaen puhuttu etenkin ketteristä ohjelmistokehitysmenetelmistä, jotka ovat tuoneet alalle joustavuutta ja tehostaneet tietojärjestelmien kehitysprosesseja. Ketterät menetelmät mahdollistavat nopean reagoimisen muuttuviin liiketoimintaympäristöihin (Highsmith 2002). Tämä edesauttaa yritysten tavoitteita saavuttaa globalisaation tuomat muutostarpeet, sillä ketterien menetelmien ansiosta tietojärjestelmien kehityksessä voidaan keskittyä asiakkaan kannalta olennaisiin toiminnallisuuksiin, joiden toiminnasta saadaan nopean kehityksen ansiosta tärkeää käyttäjäpalautetta (Boehm 2002). Tässä diplomityössä käydään läpi ketterien ohjelmistokehitysmenetelmien yleiset periaatteet. Luvussa 2 tutustutaan tarkemmin Scrumiin, joka on yleisimmän käytössä olevia ketteriä menetelmiä. Scrumin lisäksi käydään läpi Lean-ajattelutavan ja Kanban-menetelmän periaatteita. Luvussa vertaillaan lisäksi Scrumin ja Kanbanin yhtäläisyyksiä ja eroja, sekä tehdään katsaus LeSS-viitekehykseen, joka kuvaa Scrum-menetelmän skaalaamista useamman tiimin väliseen yhteistyöhön.

Diplomityössä suoritetaan havainnointiin perustuva, deskriptiivinen tapaustutkimus ketterien menetelmien käyttötavoista usean järjestelmätoimittajan välisessä tietojärjestelmähankkeessa. Hankkeessa korvataan kansainvälisen asiakkaan, nykyisin käytössä oleva, vanhentunut ERP-kokonaisuus, toteuttamalla useasta komponentista koostuva nykyaikainen tietojärjestelmäratkaisu, joka tukee pitkällä tähtäimellä yrityksen tulevaisuuden kasvutavoitteita ja liiketoiminnallisia tarpeita. Usean osapuolen välinen yhteistoiminta asettaa erityisiä vaatimuksia projektinhallinnalle ja kehitystyön ohjaamiselle. Diplomityön tavoitteena on vastata tutkimuskysymykseen: Miten ketterät ohjelmistokehitysmenetelmät soveltuvat usean järjestelmätoimittajan väliseen tietojärjestelmähankkeeseen? Tutkimuksen yhteydessä otetaan kantaa muun muassa seuraaviin kysymyksiin: Mitä ketteriä menetelmiä hankkeessa käytetään? Millä tavalla näitä menetelmiä käytetään? Mitä ketterien menetelmien käytäntöjä hankkeessa hyödynnetään? Työssä

otetaan kantaa ketterien menetelmien tuomiin hyötyihin, ja niiden asettamiin haasteisiin usean osapuolen välisessä yhteistyössä.

Tapaustutkimuksen aineistona toimii toukokuussa 2014 aloitettu, kirjoitushetkellä edelleen käynnissä oleva tietojärjestelmähanke. Kirjoittaja on toiminut hankkeessa yhden järjestelmätoimittajan projektipäällikkönä ja osallistunut hankkeen aikana yli 200 projektitapaamiseen. Tutkimuksen taustat, tavoitteet, tutkimusmenetelmät ja tapaustutkimus käydään läpi luvussa 3. Ketterien menetelmien soveltuvuutta kuvattuun hankkeeseen arvioidaan luvussa 4, käymällä läpi tutkimuksessa saavutetut havainnot ja tulokset. Tuloksien luotettavuuden ja käyttöarvon analysoinnin lisäksi, tuloksista johdetaan jatkotutkimuskohteita ketterien menetelmien hyödyntämisestä suurissa projektikokonaisuuksissa.

2. KETTERÄT OHJELMISTOKEHITYSMENETELMÄT

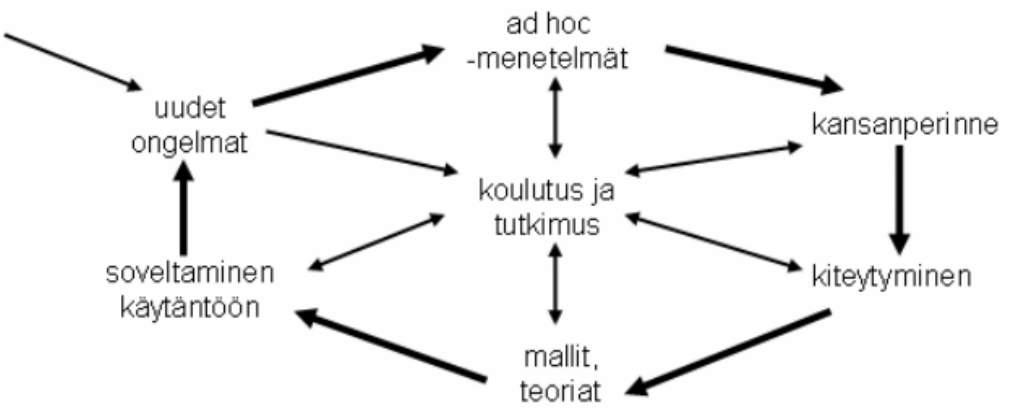
Tässä luvussa esitetään ja verrataan ketterien ohjelmistokehitysmenetelmien periaatteita. Luvussa tutustutaan ohjelmistokehityksen taustaan, jotta voidaan muodostaa näkemys ketterien menetelmien syntymiselle ja käyttötarpeelle. Luvun sisältö luo diplomityölle teoreettisen taustan, joka mahdollistaa työn tutkimuksen toteuttamisen ja arvioinnin.

2.1 Ohjelmistokehityksen tausta

Useimmat ihmiset maailmassa hyödyntävät nykyään erilaisia tietojärjestelmiä, joko tietoisesti tai tiedostamattaan. Etenkin kehittyneissä maissa ohjelmistotekniikka on perustana suurelle osalle arkisia toimiamme: Kaupassa asiointi ei olisi mahdollista ilman maksukorttia tai pankkitililtä nostettua käteistä rahaa. Sanoma- ja aikakausilehtien lukeminen painetulta paperilta, ihmisten liikkuminen pidempiä matkoja ja ravinnon valmistaminen keittiön monipuolisilla kodinkoneilla on nykypäivää hyvin pitkälti tietojärjestelmien ansiosta.

Ohjelmistotekniikan tai ohjelmistotuotannon historia ulottuu 1960-luvulle, jolloin automaattinen tietojenkäsittely (atk) otettiin terminä käyttöön. Viimeisten vuosikymmenien ja vuosien aikana ohjelmistojen käyttäminen on laajentunut huomattavasti globalisaation myötä. Tämän kasvun myötä, ja sen hallitsemiseksi, 1960- ja 1970-luvuilla alettiin kehittää prosesseja ja malleja, joilla ohjelmien tuotantoa voitaisiin hallita paremmin. (Haikala & Märijärvi 2006) Wirgentiuksen (2003) mukaan ensimmäiset ohjelmistokehitysmenetelmät syntyivät muista insinööritieteistä haettujen menetelmämallien pohjalta (katso Huttunen 2006).

Ohjelmistotuotannon kehittymisen haasteena on ollut se, että alaan liittyvä teknologia on kehittynyt niin nopeasti, että ohjelmistojen koko on voitu kaksinkertaistaa muutamien vuosien välein. Ohjelmistotekniikassa ollaan monilla osa-alueilla vasta luomassa kansanperinnettä, jolle on luonteenomaista sen ihmisriippuvuus. Ohjelmistotekniikan kehittymistä voidaan kuvata seuraavan kuvan mukaisesti. (Haikala & Märijärvi 2006)



Kuva 1. *Tieteenalojen, muun muassa ohjelmistotekniikan jatkuva kehittyminen (Haikala & Märijärvi 2006).*

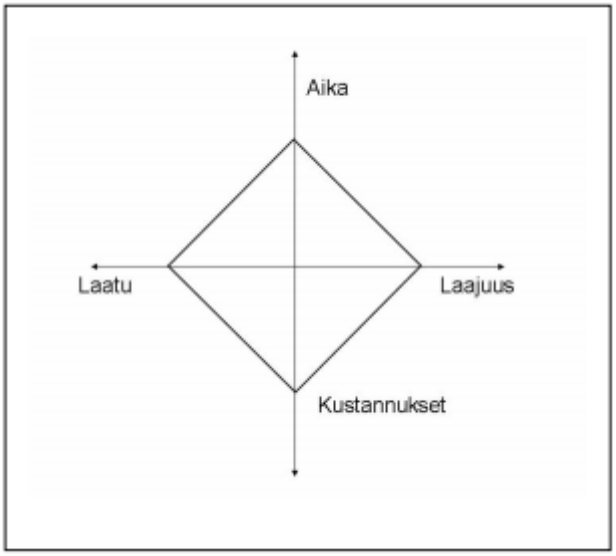
Haikala ja Märijärvi (2006) kuvaavat tämän kehittymisen vaiheita seuraavasti:

- Aluksi ongelman ratkaisuksi kelpaa mikä tahansa toimiva tapa. Sovelletaan niin sanottuja ad hoc -menetelmiä.
- Vähitellen löydetään ratkaisuja, jotka toimivat useammassa kuin yhdessä tapauksessa. Tällöin syntyy alan kansanperinnettä.
- Kun kansanperinteenä välittyvä tietotaito kehittyy systemaattisemmaksi, se kiteytetään heuristiikoiksi ja työmenetelmiksi.
- Vähitellen nämä menetelmät kehittyvät riittävän järjestelmällisesti muodostaakseen toimivia malleja ja teorioita.
- Kun näitä uusia malleja ja teorioita sovelletaan käytäntöön, löydetään uusia ongelmia ja kokonaisia sovellusalueita, joille entiset ratkaisumallit eivät enää sovi. Näin alan kehittyminen palaa alkupisteeseensä ja on jälleen lähdettävä kehittämään uusia malleja ad hoc -menetelmiä hyödyntäen.

Kansanperinteen luonteen vuoksi ohjelmistoprojektit ja niiden tuotoksena valmistuvat sovellukset ovat vain yhtä hyviä kuin niitä tekevät ihmiset ovat. Tuottavuus ihmisten välillä voi vaihdella hyvinkin paljon. Mallien ja teorioiden, sekä käytännön pohjalta muodostuneiden ohjelmistokehitysmenetelmien avulla pyritään saavuttamaan alan haasteista riippumatta kelvollinen lopputulos. (Haikala & Märijärvi 2006; Huttunen 2006) On syytä tarkentaa, että ohjelmistokehitysmenetelmä sisältää varsinaisen ohjelmistototeutustyön ohella myös ohjeita ihmisten väliseen kommunikaatioon ja yhteistyöhön, aikataulujen hallintaan ja lopputuloksen laadunvarmistukseen (Kalermo & Rissanen 2002).

Ohjelmistokehitysmenetelmän valintaan vaikuttavat samat yleiset tekijät, jotka ohjaavat myös muuta projektimuotoista toimintaa. Tekijät voidaan esittää kuvan 2 mukaisella nelikenttämallilla, joka sisältää seuraavat tekijät: laajuus, laatu, aika sekä kustannus. Ohjelmistokehitysprojektissa asiakkaan tarpeet ja projektin luonne määrittelevät sen,

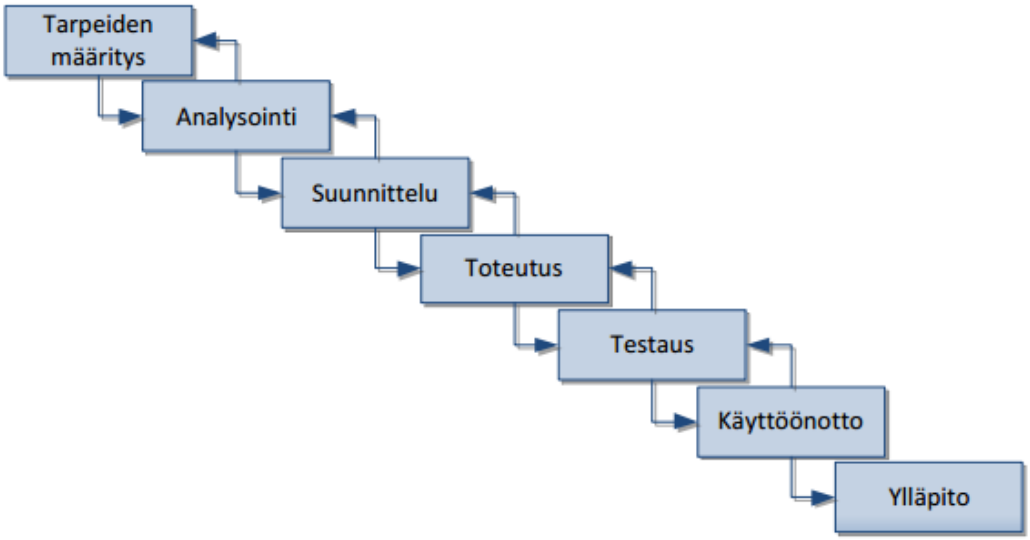
mitä tekijöitä projektissa on syytä korostaa, ja millainen kehitysmenetelmä on projektiin sopivin, jotta toivotut lopputulokset saavutetaan. (Lindberg 2003) Käytettävän ohjelmistokehitysmenetelmän valinta on joka tapauksessa aina asiakkaan ja ohjelmiston toimittavan osapuolen yhteisesti sovittava.



Kuva 2. *Ohjelmistokehitysprojekteja ohjaavat tekijät (Lindberg 2003).*

Ohjelmistokehitysmenetelmiä on ajan kuluessa muodostunut useita uusien mallien ja teorioiden yleistymisen myötä. Yleisesti menetelmät voidaan jakaa pääosin kahteen kategoriaan: perinteisiin, suunnitelmalähtöisiin ohjelmistokehitysmenetelmiin sekä ketteriin ohjelmistokehitysmenetelmiin. Jokainen menetelmä ottaa yleensä kantaa sekä ohjelmistoprojektin hallinnan näkökulmiin että ohjelmistokehityksen käytäntöihin ja sääntöihin. (Todman 2001; Hällström 2011)

Suunnitelmalähtöiset ohjelmistokehitysmenetelmät noudattavat yleisesti ohjelmistokehityksen alkuaikoina tunnistettuja toimintatapoja. Näihin toimintatapoihin kuuluu selkeän ja tarkan suunnitelman ja määrittelyn tekeminen. Tämän tavoitteena on se, että varsinaisessa tietojärjestelmän toteutusvaiheessa, ei tule vastaan kysymyksiä aiheuttavia epäselviä toteutustehtäviä. Kaikki suunnitelmat pyritään tekemään niin tarkoiksi, että toteutusvaihe etenee niiden mukaisesti ilman häiriöitä. Vaihejakomalli on yleisimmin käytetty suunnitelmalähtöinen menetelmä. Nimensä mukaisesti tässä mallissa tietojärjestelmän elinkaari jaetaan vaiheisiin. Elinkaarella tarkoitetaan aikaa, joka ohjelmistoprojektin aloittamisesta kuluu ohjelmiston poistamiseen käytöstä. Kuva 3 esittää suosituimman vaihejakomallin, vesiputousmallin rakenteen. Vesiputousmallissa ensimmäinen vaihe on tarpeiden määritys, jonka jälkeen näiden pohjalta tehdään tarkat suunnitelmat toteutustyötä varten. Suunnitteluvaiheesta voidaan palata takaisin tarpeiden analysointiin, jos suunnittelun yhteydessä edellisen vaiheen tuloksissa havaitaan tarkennettavia asioita. Tätä vaiheistettua etenemistä jatketaan sovelluksen ylläpitovaiheeseen saakka. (Haikala & Märijärvi, 2006)



Kuva 3. *Esimerkki vesiputousmallista (Haikala & Märijärvi 2006).*

Suunnitelmalähtöisistä menetelmistä poiketen, ketterät menetelmät perustuvat iteraatioihin ja inkrementaaliseen kehitykseen. Tämä tarkoittaa sitä, että toimintoja kehitetään vaiheittain aloittaen yksittäisestä toiminnosta, jota laajennetaan myöhemmissä vaiheissa kattamaan toiminnolle määritellyt vaatimukset. Tälle lähestymistavalle on tyypillistä se, että määrittelyt ja suunnitelmat muuttuvat projektin edetessä, ja niihin pystytään reagoimaan sovittujen toimintatapojen mukaisesti. (Haikala & Märijärvi 2006) Seuraavissa luvuissa käydään läpi tarkemmin ketterien menetelmien periaatteita ja käytäntöjä, sekä tutustutaan tarkemmin yleisimpiin ketteriin menetelmiin.

2.2 Ketterien ohjelmistokehitysmenetelmien yleiset periaatteet

1990-luvun loppupuolella ohjelmistotuotantoon alkoi syntyä useita kevyiksi menetelmiksi kutsuttuja ohjelmistokehitysmenetelmiä. Nämä menetelmät toivat ohjelmistokehitykseen uudenlaisen toimintatavan, jossa vallitsi läheinen yhteistyö ohjelmistokehittäjien ja liiketoiminnan ammattilaisten välillä. Kommunikointia käytiin usein kasvokkain, ja asiakkaalle tuotettiin säännöllisin väliajoin uusia toimintoja tämän investointiin liittyen. Osapuolten välillä sovittiin ohjelmointikäytännöistä ja tiimirakenteista, jotka mahdollistivat nopean reagoinnin asiakkaan muuttuviin liiketoimintavaatimuksiin. (Kosonen 2005, Huttunen 2006)

Termi ketterät ohjelmistokehitysmenetelmät syntyi vuonna 2001, kun näiden menetelmien merkittävät kehittäjät kokoontuivat yhteiseen työpajaan pohtimaan menetelmiensä välisiä yhtäläisyyksiä. Tapaamisen tuloksena julkaistiin ketterän ohjelmistokehityksen

julistus, *"Agile Manifesto"*, joka kuvaa ketterän ohjelmistokehityksen yleiset arvot ja periaatteet. (Beck et al. 2001; Kosonen 2005; Abrahamsson et al. 2002).

Julistuksen mukaisesti kaikkien ketterien menetelmien lähtökohtaisena perustana on noudattaa sen määrittelemiä arvoja, sekä yleisiä periaatteita (Beck et al. 2001). Julistus on esitetty kokonaisuudessaan alkuperäismuodossa liitteessä 1. Julistus määrittelee ketterän ohjelmistokehityksen perusarvot, joiden mukaisesti ketterissä menetelmissä arvostetaan:

- Yksilöitä ja kanssakäymistä enemmän kuin menetelmiä ja työkaluja
- Toimivaa ohjelmistoa enemmän kuin kattavaa dokumentaatiota
- Asiakasyhteistyötä enemmän kuin sopimusneuvotteluja
- Vastaamista muutokseen enemmän kuin pitäytymistä suunnitelmassa

Julistuksen mukaisesti, tämä ei kuitenkaan tarkoita, että jälkimmäiset asiat voi jättää kokonaan huomiotta. Myös niillä on arvoa, mutta vähemmän kuin ensiksi mainituilla.

Menetelmien perusarvojen lisäksi julistus esittää 12 periaatetta, joita ketterä ohjelmistokehitys noudattaa (Beck et al. 2001). Ne voidaan suomentaa seuraavasti:

1. Tärkein tavoite on pitää asiakas tyytyväisenä toimittamalla nopeasti ja jatkuvasti tarpeet täyttäviä versioita ohjelmistosta.
2. Ketterät menetelmät mahdollistavat asiakkaan kilpailukyvyn edistämisen hyödyntämällä muutosta. Muuttuvat vaatimukset ovat kehitykseen tervetulleita, myös kehityksen myöhäisessä vaiheessa.
3. Ohjelmistosta toimitetaan toimivia versioita säännöllisesti, parin viikon tai kuukauden välein, suosien lyhyempää aikaväliä.
4. Liiketoiminnan edustajien ja ohjelmistokehittäjien on toimittava yhdessä päivittäin koko projektin ajan.
5. Projektit on rakennettava motivoituneiden yksilöiden ympärille, antaen heille puitteet ja tuen, joita he tarvitsevat saadakseen työn tehtyä.
6. Kasvokkain käytävä keskustelu on tehokkain ja toimivin tapa tiedon välittämiseksi kehitystiimille ja tiimin jäsenten välillä.
7. Toimiva ohjelmisto on ensisijainen edistymisen mittari.
8. Ketterät menetelmät kannustavat kestävään toimintatapaan. Hankkeen omistajien, kehittäjien ja ohjelmiston käyttäjien tulisi pystyä ylläpitämään työtahtinsa nyt ja tulevaisuudessa.
9. Jatkuva tekniseen laatuun ja hyvään ohjelmiston rakenteeseen keskittyminen edesauttaa ketteryyttä.
10. Yksinkertaisuus on oleellista. Tekemättä jätettävä työ on pyrittävä maksimoimaan.
11. Itseohjautuvuus takaa parhaiden arkkitehtuurien, vaatimusten ja suunnitelmien kehittymisen tiimeissä, jotka organisoivat itse toimintansa.

12. Tiimin toiminnan säännöllinen tarkastelu auttaa tiimiä parantamaan tehokkuuttaan mukauttamalla toimintaansa.

Abrahamsson (2002) tiivistää edellä kuvatut arvot ja yleiset periaatteet ketterien ohjelmistokehitysmenetelmien tunnusmerkiksi seuraavasti: ketterä ohjelmistokehitys on inkrementaalista, yhteistyössä tapahtuvaa, suoraviivaista ja muutokseen sopeutuvaa (katso Huttunen 2006). Poimala (2013) puolestaan tiivistää ketterien menetelmien yleiset periaatteet käytännön ohjelmistokehityksessä viiteen kohtaan:

- Ennakkomäärittelyä ei ole tarpeen tehdä liikaa.
- Toteutus kannattaa aloittaa vaikeimmista ja tärkeimmistä asioista.
- Ohjelmistoa ei kehitetä osissa, vaan jatkuvasti iteratiivisesti parantaen ja testaten.
- Testaus on jatkuva toiminto toteutuksen yhteydessä, ei vain erillinen vaihe projektin lopussa.
- Käyttäjätarinat ovat teknisiä yksityiskohtia tärkeämpiä määrittelyä tehtäessä.

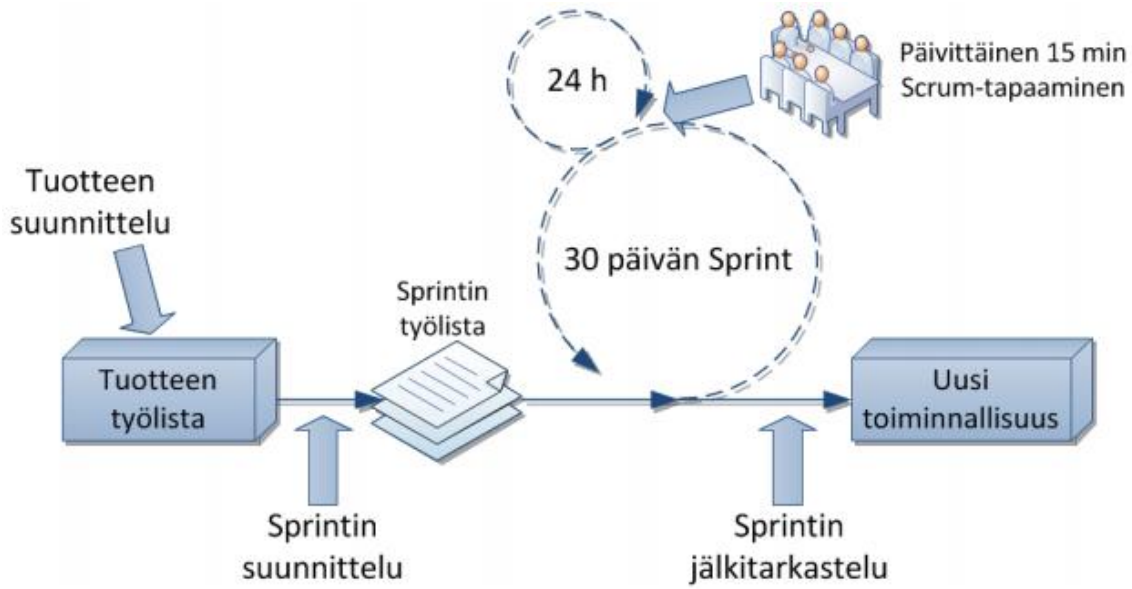
Ketteriä ohjelmistokehitysmenetelmiä on useita erilaisia. Niiden yhteisinä piirteinä ovat useimmiten iteratiivinen ohjelmistokehitys ja tiivis yhteistyö asiakkaan kanssa. Menetelmien eroja ovat niiden keskittyminen ohjelmiston eri elinkaaren vaiheisiin tai niiden ohjeistuksen tarkkuus käytännön soveltamisen näkökulmasta. Tietyt menetelmät korostavat konkreettisia käytäntöjä ohjelmointityöhön, kun taas toiset voivat olla hyvinkin abstrakteja. (Liedenpohja 2006; Huttunen 2006) Seuraavassa esitellään ketteristä menetelmistä yleisimmät, Scrum ja Kanban, sekä Lean-ajattelutapa. Tämän työn tutkimus keskittyy nimenomaan näihin kolmeen näkökulmaan.

2.3 Scrum

Scrum on yksi yleisimmistä käytössä olevista ketterän ohjelmistokehityksen menetelmistä. Scrum keskittyy ennen kaikkea projektin vaiheistamiseen ja projektin myötä syntyvän tietojärjestelmän jatkuvaan, inkrementaaliseen kehitykseen. Scrumissa tiimi työskentelee itseohjautuvasti yhteisiä projektin asettamia tavoitteita kohti. Menetelmä ei ota kantaa ohjelmiston varsinaiseen kehittämistyöhön, vaan sen avulla hallitaan projektin kehityksen kokonaisuutta. Scrum sisältää monia käsitteitä kuten tiimin jäsenten välisiä rooleja, säännöllisiä tapaamisia tiimin sisällä ja asiakkaan kanssa, iteratiivisesti saavutettavia tuloksia, sekä toimintatapoja projektin aikaiseen työskentelyyn. (Schwaber & Beedle 2002; Vuorinen 2011)

Scrum-menetelmän perusteoksena voidaan pitää Schwaberin ja Beedlen (2002) kirjoittamaa *"Agile Software Development with Scrum"* -teosta. Tässä luvussa ja seuraavissa aliluvuissa esitettävät Scrumin periaatteet ja käytännöt perustuvat Schwaberin ja Beedlen (2002) teokseen.

Kuvassa 4 esitetään Scrum-menetelmän keskeiset käsitteet ja vaiheistusmalli. Scrumissa projekti muodostuu tuotteen työlistasta (*"Product backlog"*), joka sisältää kaikki ne toiminnallisuudet ja tehtävät, jotka projektissa on tarkoitus toteuttaa. Projekti etenee iteratiivisesti vaiheissa, joita kutsutaan sprinteiksi. Sprintin kesto on yleensä yksi kuukausi ja sprintin työlista (*"Sprint backlog"*) määrittää kunkin kuukauden aikana toteutettavan osasisällön tuotteen työlistasta. Sprintin suunnittelussa (*"Sprint planning meeting"*) tuotteen työlistasta otetaan tietty joukko toiminnallisuuksia ja tehtäviä, joista muodostuu sprintin työlista. Sprintin edistymistä seurataan lyhyillä, päivittäisillä Scrum-tapaamisilla (*"Daily Scrum"*), joiden aikana jokainen tiimin jäsen käy läpi hänen edellisen ja tulevan päivän tehtävät, sekä tuo ilmi mahdolliset ongelmat, jotka vaikeuttavat tai estävät hänen tehtäviensä suorittamista. Sprintin jälkeen kuukauden aikana tehdyt tehtävät käydään läpi sprintin jälkitarkastelussa (*"Sprint review meeting"*) ja näistä syntyy uutta toiminnallisuutta projektin lopputulokseen. Tätä iteratiivista prosessia jatketaan kunnes sovelluksen päätetään toteuttavan kaikki vaaditut toiminnallisuudet.



Kuva 4. *Scrum-menetelmän keskeiset käsitteet ja vaiheistus (Schwaber et al. 2005; af Hällström 2011).*

Scrum-menetelmä perustuu empiiriseen prosessikontrolliteoriaan, jonka mukaan menetelmässä oleellisena elementtinä on jokaisen iteraation päätteeksi annettava jatkuva palaute ja sen käsittely. Tämän toimintatavan tavoitteena on parantaa projektin lopputulosta sekä tuotteen että prosessin näkökulmasta. Palautteen läpikäyminen ja prosessin kriittinen tarkastelu mahdollistavat jatkossa sopeutumisen uusiin tilanteisiin ja projektitiimin toimintatapojen kehittämisen. (Schwaber 2004)

2.3.1 Scrum-tiimin rakenne ja roolit

Scrum-menetelmässä keskeinen tekijä on tiimi, joka on valittu toteuttamaan menetelmän mukaisesti projektia. Tiimi koostuu henkilöistä, joilla on erilaiset roolit ja vastuu-

alueet. Tuoteomistaja (*"Product owner"*) vastaa projektin kannattavuudesta priorisoimalla tuotteen työlistan. Tuoteomistaja on läheisessä yhteydessä projektin sidosryhmiin, joihin kuuluvat muun muassa asiakas, järjestelmätoimittajat, sovelluksen käyttäjät, projektin rahoittajat sekä muut projektin vaikutuspiirissä olevat osapuolet. Tuoteomistaja ohjaa jokaisen sprintin aluksi tehtävää suunnittelua ja hyväksyy edellisen sprintin tulokset.

"ScrumMaster" on henkilö, joka vastaa Scrum-menetelmän prosessien noudattamisesta ja tiimin tuottavuudesta. ScrumMaster auttaa tiimiä työskentelemään yhteisen tavoitteen eteen, poistaa esteet, jotka haittaavat tiimin työskentelyä, sekä mahdollistaa tiimin rauhallisen ja häiriöttömän työskentelyn. Erityisesti häiriöiden poistaminen on tärkeää, sillä tiimin tuottavuus ja tehokkuus heikkenevät jatkuvien keskeytysten vuoksi. Vaikka ScrumMasterin tehtävänä on valvoa menetelmän mukaisien prosessien noudattamisesta, hänellä ei kuitenkaan ole tiimiin nähden määräysvaltaa, sillä tiimi on itseohjautuva ja omat päätöksensä tekevä ryhmä ihmisiä. Tiimi on siis vastuussa itsensä johtamisesta. Tiimin pääasiallisena tehtävänä on toteuttaa sprintin suunnitelman mukaiset toiminnot ja tehtävät. Näistä yksittäisien sprinttien tuloksista muodostuu lopulta projektissa toteutettava kokonaisuus inkrementaalisesti kasvaen. Scrum-tiimille määritelty yleinen ja toimiva koko on 5-9 henkilöä. Tiimin itseohjautuvuus tarkoittaa sitä, että tiimissä ei ole selkeää tehtäväjakoa toteutustyön suhteen, vaan tiimi itse suunnittelee millä tavalla se organisoituu toteuttamaan sprintin työlistan sisältämät tehtävät.

2.3.2 Scrumin käytännöt

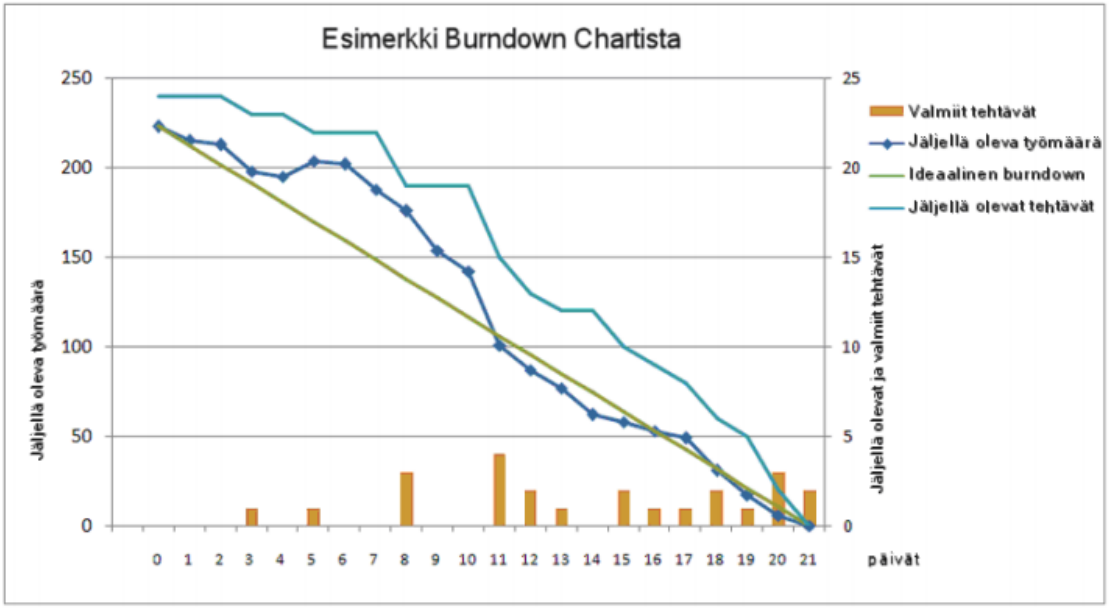
Scrumin iteratiivisuuden johdosta menetelmälle on tyypillistä jatkuva suunnittelu, sekä toimintatapojen arviointi ja tehostaminen. Tiimin itseohjautuvuus antaa tiimille suuren vastuun projektin edistymisestä, sillä tiimi arvioi itse jokaisen sprintin suunnittelun yhteydessä miten paljon toiminnallisuuksia ja tehtäviä on sprintin aikana mahdollista toteuttaa, käytettävissä oleva aika ja resurssit huomioiden. Scrumissa työmäärän arviointiin käytetään pokerisuunnittelua (*"Poker planning"*). Pokerisuunnittelussa jokainen sprintin työlistalla oleva tehtävä käydään yksitellen läpi, ja jokainen tiimin jäsen arvioi oman työmääränsä kyseisen tehtävän toteuttamiseen tähän tarkoitukseen suunniteltuilla pelikorteilla. Pelikortit sisältävät pistemääriä, jotka suhteuttavat tehtävien viemän työmäärän toisiinsa. Tiimin jäsenten esitettyä arvionsa tehtävän suorituksesta, pyritään löytämään yhteisymmärrys vaaditusta työmäärästä. Suunnittelun ideana on, että jos tiimin arviot työmäärästä eroavat toisistaan, pienimmän ja suurimman arvion antaneet henkilöt perustelevat näkemyksensä työmäärästä, minkä jälkeen arviointi tehdään uudelleen ja tiimi voi ottaa huomioon nämä näkemykset uudessa arviossaan. Pokerisuunnittelun tavoitteena on tehdä suuntaa-antava estimointi tehtävien työmääristä nopeasti ja tehokkaasti sprintin suunnittelun helpottamiseksi.

Tiimin tehtävänä on myös suunnitella kuinka sprintin tehtävät toteutetaan yksityiskohtaisesti. Tavoitteena on saavuttaa valmiita toiminnallisuuksia jokaisen sprintin aikana.

Valmiin toiminnallisuuden määritelmä voi vaihdella ja tiimin tehtävänä on sopia se yhdessä projektin muiden sidosryhmien kanssa. Toiminnallisuuden määrittelyssä käytetään usein käyttäjätarinaa (*"User story"*), joka kuvaa loppukäyttäjän sovelluksella tekemää toimintoa. Käyttäjätarina kertoo hyvin selkeästi ne konkreettiset toimenpiteet, joita käyttäjä tekee sovellusta käyttäessään. Tarinoiden avulla sovelluksen toiminnallisuuksille saadaan nopeasti kuvattua vaatimukset, jotka tiimin tulee sovellukseen toteuttaa. Käyttäjätarinat voivat sisältyä tuotteen työlistaan. Sprintin suunnittelun myötä tiimi sitoutuu toteuttamaan suunnittelemansa toiminnallisuudet ja tehtävät, sekä täyttämällä näihin liittyvät vaatimukset arvioimansa työmäärän mukaisesti.

Päivittäisillä Scrum-tapaamisilla seurataan sprintin edistymistä yhdessä tiimin ja ScrumMasterin kanssa. Tapaamiset ovat lyhyitä ja tehokkaita, ja niiden tarkoituksena on antaa selkeä kuva koko tiimille mitä tehtäviä kukin tiimin jäsen kyseisenä päivänä tekee. Tapaamisen tarkoituksena on myös auttaa muita tiimin jäseniä heidän tehtävissään käymällä läpi tehtäviä hidastavat ja vaikeuttavat ongelmat, joiden selvittämisessä tiimin muut jäsenet voivat auttaa.

Burndown-kaavio (*"Burndown chart"*) havainnollistaa sprintin, julkaisun tai kokonaisprojektin jäljellä olevaa työmäärää. Kaavio muodostuu sprintin tai tuotteen työlistasta, ja se esittää sprintin suunnittelussa arvioidun työmäärän suhteessa sprintin jäljellä olevaan aikaan ja suunniteltuihin tehtäviin, jotka sprintin työlistalle tiimin toimesta valittiin. Kaaviota on tarkoitus päivittää päivittäin, jotta siitä nähdään helposti sprintin etenemisen tilanne. Ajan tasalla olevasta burndown-kaaviosta nähdään jatkuvasti onko tiimi toteutusmäärältään edellä vai jäljessä aikataulua, jotta jäljellä olevan sprintin aikana ehditään toteuttamaan kaikki suunnitellut tehtävät. Kaavio kuvaa tiimin nopeutta (*"Velocity"*), ja sitä kuinka paljon tiimi pystyy sprintin aikana tekemään tehtäviä. Burndown-kaaviota tarkastelemalla tulevia sprinttejä voidaan suunnitella tehokkaammin ja realistisemmin, kun kaaviosta nähdään toteutunut ajankäyttö suhteessa arvioituun ajankäytön tarpeeseen sprintin toteutuksen osalta.



Kuva 5. *Esimerkki Burndown-kaaviosta (Burn down chart 2015; katso Vuorinen 2011).*

Sprintin lopuksi tehdään sprintin jälkitarkastelu tai katselmointi, jonka aikana tiimi käy läpi sprintin aikana toteutetut toiminnallisuudet ja tehtävät. Tapaamisessa ovat tiimin lisäksi mukana tuoteomistaja ja mahdollisesti projektin muiden sidosryhmien edustajia. Katselmoinnin tarkoituksena on esittää vain täysin valmiita toiminnallisuuksia. Sprintin retrospektiivi (*"Retrospective meeting"*) on tiimin ja ScrumMasterin välinen sprintin päätöstilaisuus, jonka tarkoituksena on käydä läpi asioita, joita seuraavassa sprintissä olisi tarpeen muuttaa, jotta työ olisi tuottavampaa ja miellyttävämpää tiimin jäsenille.

Kokonaisuudessaan projekti voi koostua useasta erillisestä Scrum-tiimistä, jotka toimivat yhteistyössä projektin lopputuloksen saavuttamiseen toteuttamalla omia sprinttejään tuotteen työlistasta. Useamman Scrum-tiimin työskentely samassa projektissa mahdollistaa Scrum-menetelmän skaalaamisen isompien projektikokonaisuuksien toteuttamiseen. Tällaista useiden tiimien välistä työskentelyä seurataan *"Scrum-of-Scrums"* -tapaamisilla. Tähän tapaamiseen osallistuu yksi henkilö jokaisesta Scrum-tiimistä. Tapaamisen tarkoituksena on vastata samoihin kysymyksiin kuin tiimien omissa päivätapaamisissa, mutta tällä kertaa käsiteltävänä kokonaisuutena on jokaisen tiimin tehtävien tilanne. *"Scrum-of-Scrums"* muodostaa eri tiimien välille laajemman kokonaiskuvan koko projektin, ja sen yksittäisten tiimien edistymisestä ja mahdollisista ongelmista.

Scrum-menetelmälle on tyypillistä, että tuotteen työlista elää ja muuttuu projektin aikana asiakkaan liiketoimintavaatimusten tarkentuessa tai muuttuessa. Tuotteen työlista on priorisoitu ja se sisältää kaikki projektin toiminnalliset vaatimukset, mutta myös muut tehtävät, joihin tiedetään kuluvan aikaa. Tuoteomistaja on vastuussa projektin liiketoiminnallisesta kannattavuudesta ja pyrkii tämän perusteella priorisoimaan tuotteen työlistan optimaalisesti, jotta projektin kannattavuustavoitteet saavutetaan. Muuttuvien

vaatimusten vuoksi Scrum-menetelmän mukaisten tapaamisten ja suunnittelupalaverien merkitys kasvaa, jotta muutoksien aiheuttamat tehtävät saadaan tiimien tietoon mahdollisimman tehokkaasti, mutta kuitenkaan häiritsemättä sen hetkistä toteutustyötä.

Scrumin tarkassa noudattamisessa on edellä kuvattujen käytäntöjen lisäksi määritelty hyvinkin tarkkoja toimintaohjeita esimerkiksi tapaamisten pituudesta, niihin valmistautumiseen sallitusta ajasta, sekä eri roolien vastuista kirjata tapaamisista muistiinpanoja. Scrumin ohjeistus ottaa kantaa myös siihen miten toimitaan tilanteessa, jossa sprintin jälkitarkastelussa havaitaan joidenkin tehtävien olevan keskeneräisiä, ja näin ollen sprintin työlistalla olevia tehtäviä ei ole toteutettu kokonaisuudessaan. Näitä tarkkoja ohjeistuksia ei tässä yhteydessä tarkemmin käsitelty, vaan keskityttiin Scrum-menetelmän pääpiirteiden kuvaamiseen. Tämän diplomityön tutkimuksen yhteydessä käsitellään osaltaan myös näitä tarkempia ohjeistuksia.

2.4 Lean ja Kanban

Lean on johtamisen filosofia, jossa pyritään jatkuvasti kehittämään toimintaa, sekä poistamaan tuottamaton toiminta. Lean-tuotantokonsepti syntyi alun perin autoteollisuuden tarpeisiin, kun Toyota lanseerasi käyttöönsä TPS-periaatteen (*"Toyota Production System"*). Lean esiteltiin terminä 1990-luvulla, kun Womack et.al. (1991) kirjoittivat Toyotan menestyksestä autoteollisuuden prosessien muokkaajana kirjassaan *"The Machine That Changed the World"*. (katso Hirvonen 2012)

Lean-ohjelmistokehitys on Lean-filosofian soveltamista ohjelmistojen kehitykseen. Anderson (2010) ja Poppendieck & Poppendieck (2006) kuvaavat Lean-ohjelmistokehityksen periaatteet seuraavasti (katso Lehtonen et al. 2014):

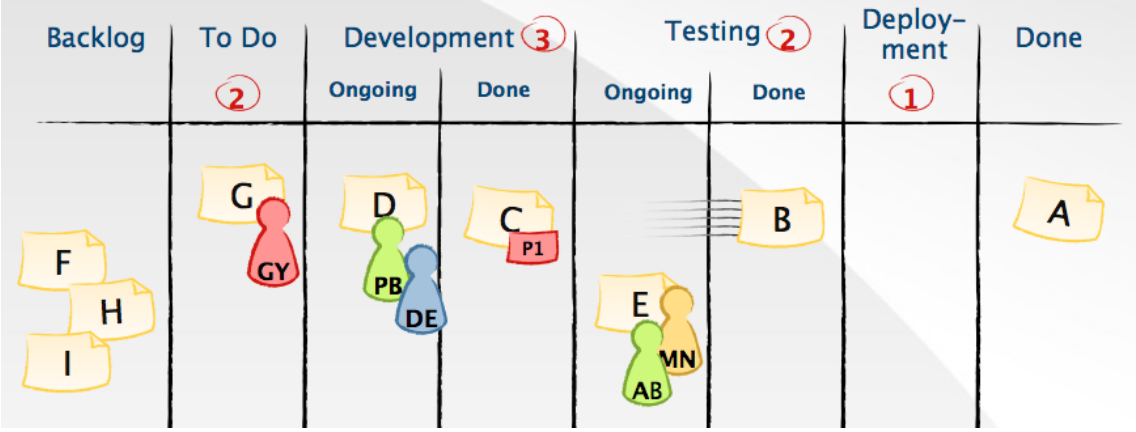
1. Poista hukka: älä tee ominaisuuksia varmuuden vuoksi, varmista nopea toteutus vaatimuksesta testaukseen
2. Rakenna laatu osaksi tuotetta: kirjoita toimivia testejä vaatimusten sijaan ja integroi jatkuvasti
3. Luo tietoa: opi uutta kokeilujen kautta, haasta nykyiset tavat ja mahdollista nopea reagointi muutoksiin
4. Lykkää sitoutumista: tee myöhäiset muutokset mahdollisiksi tekemällä rajoittavia päätöksiä vasta viime hetkellä
5. Toimita nopeasti: vältä listojen ja jonojen käyttöä ja tee toimivia versioita lyhyin väliajoin työstäen vain muutamia asioita kerrallaan
6. Kunnioita ihmisiä: kun tiimillä on mahdollisuus vaikuttaa omaan työhönsä, se on sitoutunut yhteisen tavoitteen saavuttamiseen, minkä seurauksena tiimin, johdon ja asiakkaan välillä vallitsee keskinäinen kunnioitus
7. Optimoi kokonaisuus: keskity kokonaisuuteen, toimita koko tuote ja mittaa tuloksellisuutta tuotteen hyödyllisyyden ja asiakkaan tyytyväisyyden kautta

Kanban mahdollistaa Leanin mukaisen prosessin kehittämisen organisaatiolle. Kanban ei ole varsinaisesti menetelmä, joka ohjaa tarkkojen toimintatapojen noudattamiseen, vaan Leanin tapaisesti joukko periaatteita. Kanbanin tavoitteena on optimoida arvoketju, joka on Lean-filosofian mukaisesti tehtävä näkyväksi, jotta tuottamaton toiminta saadaan tuotua esille. Kanban menetelmän avulla voidaan visualisoida kuinka arvo kulkeutuu järjestelmän läpi. Visuaalisuus on Kanbanin tärkeimpiä periaatteita. (Kniberg & Skarin 2010; Koski 2012).

Kniberg ja Skarin (2010) tiivistävät Kanbanin kolmeen pääsääntöön:

1. Työnkulun etenemisen tekeminen visuaaliseksi
2. Käynnissä olevien töiden määrän rajoittaminen (*"Work in progress"*)
3. Työn läpimenoajan mittaaminen (*"Lead time"*)

Työnkulun visualisoinnilla tarkoitetaan työn jakamista selkeisiin osiin, joiden etenemistä ja toteutusta voidaan seurata osa kerrallaan. Visualisointiin käytetään kuvan 6 mukaista Kanban-taulua, joka kuvaa sarakkeilla prosessin eri vaiheissa olevat osatyöt. Taulun avulla työnkulku on jatkuvasti näkyvillä. Käynnissä olevien töiden rajoittamisella tarkoitetaan täsmällistä tapaa merkitä jokaisessa prosessin vaiheessa sallittu yhtäaikaisten töiden määrä. Tällä estetään se, että töiden kasautuminen tiettyyn prosessin vaiheeseen, ei aiheuta hidastusta muiden vaiheiden töihin. Läpimenoajan mittaamisella kuvataan kestoa, joka yhdellä työllä kestää kulkea koko prosessin läpi. Läpimenoajan lyhentämisellä saavutetaan optimoitu prosessi, jossa työtehtävät valmistuvat parhaalla mahdollisella tehokkuudella. Läpimenoajan lyhentäminen on mahdollista mittaamalla läpimenoaikoja, sekä tekemällä pieniä muutoksia käynnissä olevien töiden määrään eri prosessin vaiheissa, ja havaitsemalla näiden muutosten vaikutukset läpimenoaikaan. Tällä prosessilla parannetaan tehokkuuden lisäksi työnkeston ennustettavuutta. (Kniberg & Skarin 2010; Lekman 2009)



Kuva 6. *Esimerkki Kanban-taulun visuaalisesta työnkulun etenemisestä (Wester 2012).*

Läpimenoajan mittaamiseen Anderson (2010) yhdistää laajemminkin työn etenemisen mittaamisen ja hallinnoinnin, jonka yhtenä osana on läpimenoajan seuranta. Andersonin (2010) mukaan tässä on keskeistä hahmottaa minkä verran käynnissä olevia töitä on prosessin eri vaiheissa. Kun Kanban-periaatteita käytetään oikein, Kanban-taulun eri vaiheiden sisällä olevien töiden määrän on tarkoitus pysyä muuttumattomana. Tällöin prosessin eri vaiheissa on jatkuvasti suunnilleen sen verran töitä, kuin kyseisen vaiheen sallituksi työmääräksi on merkitty. Kun yksittäinen työ siirtyy vaiheesta toiseen, niin tämän tilalle tulee edellisestä vaiheesta uusi työ. Näiden keinojen lisäksi työn etenemisen mittaamisessa kiinnitetään huomiota töiden suorittamiseen suhteessa niille asetettuihin eräpäiviin (*"Due date performance"*). Kanbanissa yksittäiselle työlle voidaan asettaa eräpäivä, joka kuvaa päivien määrässä aikaa, joka tämän työn läpimenoon prosessissa saa kulua. Eräpäivillä voidaan priorisoida töiden järjestystä Kanbanin työlistassa (*"Backlog"*). (Anderson 2010)

Edellä esitettyjen kolmen pääsäännön laajennukseksi Anderson (2010) tuo Kanban-periaatteeseen kaksi lisäsääntöä:

4. Prosessikäytäntöjen tekeminen täsmällisiksi
5. Valmiiden mallien käyttäminen kehitysmahdollisuuksien tunnistamiseen

Erilaisten prosessikäytäntöjen erottamiseen tyypillisin keino on eri palveluluokkiin kuuluvien töiden merkintä värikoodein tai "uimaratoja" (*"Swimlane"*) käyttämällä. Uimaratoja käytettäessä Kanban-näkymä jaetaan horisontaalisesti erillisiksi "radoiksi". Radat kuvaavat esimerkiksi toteutettavan sovelluksen toimintokokonaisuuksia. Tietyn kokonaisuuden toteuttamiseen liittyvät tehtävät käsitellään Kanban-näkymässä kyseisen uimaradan sisällä. Värikoodien käyttäminen on vaihtoehtoinen keino erottaa tehtävät toisistaan. Uimaratojen sijaan eri toimintokokonaisuudet voidaan erottaa toisistaan merkitsemällä tehtävät värikoodeilla. (Anderson 2010)

On hyvä tarkentaa, että värikoodit ja uimaradat eivät poissulje toisiaan, vaan niitä voidaan käyttää myös yhdessä. Tehtävien erottaminen värikoodeilla voidaan tehdä myös uimaratoja käytettäessä. Tämä tekee prosessikäytäntöjen erottamisesta entistä visuaalisempaa. Kanbania käytettäessä, on lisäksi tyypillistä, että tehtävät priorisoidaan. Kanban-näkymässä priorisointi esitetään siten, että tärkeimmät tehtävät ovat aina jokaisessa sarakkeessa ylimpänä, ja tehtävien merkitys pienenee alemmaksi mentäessä. (Anderson 2010)

Kanban on empiirinen menetelmä, jota käytettäessä käyttäjän oletetaan kokeilevan prosessia eri tavoin, sekä mukauttavan sen omaan projektiinsa sopivaksi. Kanban ei esimerkiksi kerro millainen rajoite käynnissä olevien töiden määrään liittyy, vaan tarkoitus on sovittaa rajoite kunkin projektin yksilöllisiin tarpeisiin. (Kniberg & Skarin 2010)

Andersonin (2010) mukaan Kanban tarjoaa sen käyttäjälle kehitysmahdollisuuksien tunnistamiseen valmiita malleja, joita voidaan hyödyntää. Nämä ohjaavat esimerkiksi tuhlauksen vähentämistä, sekä rajoitusten ja vaihtelevuuden hallintaa. Tärkeintä Kanbanin hyödyntämisessä on kuitenkin jatkuva prosessin kehittyminen tarkastelemalla sen käytössä saavutettuja tuloksia, ja mukauttamalla toimintaa näiden perusteella. Tästä seuraa prosessin tehokkuuden parantuminen. Koska Kanban on vielä suhteellisen uusi ajatus, löytyy siitä monia erilaisia määrityksiä ja ohjeistuksia (Kniberg & Skarin 2010; Koski 2012).

2.5 Scrumin ja Kanbanin yhtäläisyydet ja erot

Lean-filosofian tarkoituksena on kehittää toimintatapoja jatkuvasti, oppia tunnistamaan ja poistamaan tuottamaton toiminta. Ketterien menetelmien tavoitteena on asiakastyytyväisyyden saavuttaminen toimivan sovelluksen myötä. Ketterissä menetelmissä tähän pyritään suhteuttamalla toteutustyö, suunnittelu ja muut prosessit järkevällä tavalla. Kniberg & Skarin (2010) mukaan sekä Scrum että Kanban ovat molemmat Lean-filosofian ja ketterän ajattelutavan mukaisia. Molemmat menetelmät tavoittelevat optimaalisia toimintatapoja läpinäkyvyyden, tehtävien jakamisen, työmäärän hallinnan ja itseohjautuvien tiimien avulla. Menetelmissä keskitytään toimittamaan julkaistavia versioita sovelluksesta mahdollisimman usein. Julkaisuprosessia tehostetaan empiiristen kokemusten perusteella, joko Scrum-tiimin tehokkuuden tai Kanbanin läpimenoajan mittaamisella.

Scrum ja Kanban ovat molemmat prosessityökaluja, joiden tarkoituksena on tehostaa työskentelyä. Kumpikaan menetelmä ei ole työkalu, joka kertoo tarkalleen mihin ja miten sitä käytetään, vaan ne tarjoavat tiettyjä rajoitteita ja ohjeistuksia toimintatapoihin. Mikään työkalu ei ole täydellinen, eikä Scrum tai Kanban sovellu jokaiseen tilanteeseen. Ketterien menetelmien ja Lean-filosofian perusperiaatteiden mukaisesti, oleellista on tutustua, kokeilla, löytää ja muokata omaan tilanteeseen sopivat prosessit. Scrumin ja Kanbanin yhtäläisyyksien johdosta, niitä nähdään usein käytettävän toisiaan täydentävinä toimintapoina. (Kniberg & Skarin 2010)

Scrumin ja Kanbanin väliset eroavaisuudet on listattu taulukossa 1 Kniberg & Skarin (2010) esittämän mukaisesti. Erot eivät ole merkittäviä, ja niiden voidaan ennemminkin nähdä täydentävän toisiaan, jos menetelmiä käytetään yhdessä.

Taulukko 1. *Scrumin ja Kanbanin erot (Kniberg & Skarin 2010).*

Scrum	**Kanban**
Määrittelee ajallisesti rajatut iteraatiot.	Ajallisesti rajatut iteraatiot mahdollisia. Tapahtumapohjainen rajaus yleisempää. Suunnittelu, julkaiseminen ja prosessin kehittäminen voidaan suorittaa erillään.

Tiimi sitoutuu työmäärään iteraatiokohtaisesti.	Sitoutuminen on vapaaehtoista.
Nopeus (*"Velocity"*) on pääasiallinen suunnittelun ja prosessin tehostamisen mittari.	Läpimenoaika (*"Lead time"*) on pääasiallinen suunnittelun ja prosessin tehostamisen mittari.
Määrittelee monitoiminnalliset tiimit.	Asiantuntijatiimit ovat sallittuja, monitoiminnalliset valinnaisia.
Tehtäväkokonaisuudet täytyy pilkkoa niin pieniksi tehtäviksi, että ne voidaan suorittaa sprintin aikana.	Tehtävillä ei määriteltyä kokoa.
Määrittelee Burndown-kaavion.	Ei määriteltyjä kaavioita.
Käynnissä olevien töiden määrä on rajattu epäsuorasti, sprintin mukaiseksi.	Käynnissä olevien töiden määrä on rajattu prosessin vaiheiden mukaisesti.
Määrittelee työmäärän arvioinnin.	Työmäärän arviointi on vapaaehtoista.
Käynnissä olevaan sprinttiin ei voida lisätä uusia tehtäviä.	Uusia tehtäviä voidaan lisätä aina, kun resurssit ja käynnissä olevien töiden määrä sen sallii.
Sprintin työlistan omistaa yksi tiimi.	Kanban-näkymä voidaan jakaa usean tiimin välillä.
Määrittelee kolme roolia.	Ei määrittele rooleja.
Scrum-näkymän sisältö tyhjennetään sprinttien välillä.	Kanban-näkymä on pysyvä.
Määrittelee priorisoidun tuotteen työlistan.	Priorisointi on vapaaehtoista.

2.6 LeSS – Large-Scale Scrum

Scrum-menetelmää käytetään yleisesti yhden kehitystiimin prosessinohjaukseen. Menetelmän teorioissa ja periaatteissa se kuvataan yksittäisen tiimin käyttämäksi työkaluksi. Scrumin skaalaaminen laajempaan käyttöön on nähty haastavaksi, sillä tiimin koon kasvattaminen tekee kommunikoinnista ja prosessin noudattamisesta vaikeampaa. Tähän vaikuttaa etenkin se, että Scrumia käytettäessä ei aina tiedetä varmaksi mitä seuraavaksi tehdään, jolloin epävarmuuden organisoiminen suuremmalle tiimille on haastavaa, ja voi laskea Scrum-tiimin tehokkuutta. Erillisten ominaisuustiimien (*"Feature teams"*) käyttäminen on nähty toimivimpana keinona Scrumin skaalaamisessa. Tässä lähestymistavassa kukin tiimi kehittää tiettyä ominaisuutta tai toimintokokonaisuutta, jotka yhdessä muodostavat laajemman sovelluksen. Ominaisuustiimien käyttäminen suuressa organisaatiossa vaatii kuitenkin paljon oppimista käytännöistä, Scrumin mukaisesta läpinäkyvyydestä ja menetelmän mukaisesta kommunikoinnista. Uusien toimintatapojen opetteleminen ja omaksuminen voivat heikentää organisaation tehokkuutta. (James 2012)

Craig Larman ja Bas Vodde ovat tutkineet 2005 vuodesta alkaen Scrumin skaalaamista suurien organisaatioiden käyttöön. He ovat olleet ohjaamassa ja ottamassa käyttöön Scrum-menetelmän toimintatapoja useissa suurissa organisaatioissa kuten BMW, Nokia Networks, Ericsson ja JP Morgan Chase. Kaksikko julkaisi kokemuksiinsa pohjautuen Scrumin skaalaamisesta kaksi teosta, joita voidaan pitää tämän periaatteen perusteoksina: *"Scaling Lean & Agile Development"* (2008) ja *"Practices for Scaling Lean & Agile Development"* (2010).

Vuonna 2013 Larmanin ja Vodden kokemukset vakiinnutettiin määrittelemällä *"LeSS, Large-Scale Scrum"* -viitekehys (2014). Viitekehyksestä on tehty kaksi tasoa, organisaation kokoluokan mukaan: *"(basic) LeSS"* ja *"LeSS Huge"*. Näistä jälkimmäinen on tarkoitettu organisaatioille, joissa on yli kahdeksan erillistä Scrum-tiimiä ja jopa tuhansia henkilöitä toteuttamassa yksittäistä tuotetta. Pienempi näistä on tarkoitettu 2-8 tiimin organisaatioille. Luvut eivät ole täsmällisen tarkkoja, vaan ne riippuvat tilanteesta. Termiä *"LeSS"* käytetään yleisesti kuvaamaan *"Large-Scale Scrumin"* mukaisia viitekehyksiä. (Larman & Vodde 2014) Seuraavassa esitetään tätä termiä käyttäen pienemmän LeSS-viitekehyksen keskeiset käytännöt.

LeSS:n tarkoituksena on skaalata Scrum-menetelmä suuren organisaation käyttöön. LeSS pyrkii kohdentamaan jokaisen tiimin huomion kokonaisuuteen, sen sijaan, että tiimi keskittyy vain omaan osuuteensa. LeSS sisältää periaatteita ja sääntöjä, jotka määrittelevät viitekehyksen. LeSS tarjoaa ohjeita ja kokeiluja, jotka opastavat organisaatioita omaksumaan *"Large-Scale Scrumin"* osaksi toimintaansa. Larmanin & Vodden (2014) kuvaamat LeSS:n mukaiset 10 periaatetta ovat:

1. **Large-Scale Scrum on Scrumia**. LeSS ei ole "uusi ja parannettu Scrum", vaan ennemminkin ymmärrystä siitä, kuinka soveltaa Scrumin mukaisia periaatteita, elementtejä ja tarkoituksia laajassa kontekstissa.
2. **Empiirinen prosessinhallinta**. Tuotteen, prosessien, organisaatiorakenteen ja toimintatapojen tarkastelu ja sovittaminen määrittelevät tilanteeseen sopivan Scrumin hyödyntämismallin. Tarkkojen toimintaohjeiden asettaminen on turhaa.
3. **Läpinäkyvyys**. Empiirinen prosessinhallinta vaatii ja luo läpinäkyvyyttä. Läpinäkyvyys perustuu konkreettisiin toteutettuihin tehtäviin, lyhyisiin iteraatioihin, yhteistyöhön, yleisiin määrityksiin ja pelon poistamiseen työympäristöstä.
4. **Enemmän vähemmällä ("more with less")**. 1) Empiirisessä prosessinhallinnassa saavutetaan enemmän oppimista, pienemmällä määrällä määritettyjä prosesseja. 2) Lean-ajattelutavassa saavutetaan enemmän arvoa, vähemmällä hukalla ja turhalla työllä. 3) Skaalaamisessa on vähemmän rooleja, häiriötä ja erityisryhmiä.
5. **Fokus kokonaistuotteessa**. Yksi tuotteen työlista, yksi tuoteomistaja, yksi mahdollinen julkaistava tuoteinkrementointi, yksi sprintti - riippumatta siitä

onko projektissa kolme vai 33 tiimiä. Asiakas haluaa tuotteen, ei vain tiettyä osaa siitä.

6. **Asiakaslähtöisyys**. Havaitse asiakkaan kokema arvo, ja turhat tehtävät. Vähennä toteutukseen kuluvaa aikaa asiakkaan silmissä. Kerää paljon palautetta todellisilta käyttäjiltä. Tee kaikille tiimien jäsenille selväksi millä tavalla jokapäiväinen työ vaikuttaa ja hyödyttää asiakasta.
7. **Jatkuva kehittyminen täydellisyyden tavoittelemisessa**. Toteuta ja toimita tuotetta jatkuvasti, ilman maksua ja vikoja, mikä kerta kaikkiaan riemastuttaa asiakasta, parantaa ympäristöä ja tekee elämästä parempaa. Tee nöyriä ja perusteellisia parannuksia ja kokeiluja jokaisessa sprintissä saavuttaaksesi täydellisyyden.
8. **Ajattele kokonaisjärjestelmää**. Näe, ymmärrä ja paranna kokonaisuutta, tutki järjestelmien välistä dynamiikkaa. Asiakas välittää kokonaisuudesta ja sen nopeudesta tuottaa konseptista rahaa (“concept-to-cash cycle time”). Keskity siis kokonaisjärjestelmän tehokkuuteen ja tuottavuuteen, yksittäisten vaiheiden optimoinnin sijaan.
9. **Lean-ajattelutapa**. Tavoittele täydellisyyttä rakentamalla organisaatio, jossa kokonaisuuden perustana on “johtajat opettajina” -periaate. Tämän mukaisesti organisaation johdon täytyy noudattaa ja opettaa kokonaisuuden ajattelua sekä Lean-filosofian mukaisia periaatteita. Lisää tähän henkilöiden kunnioitus ja jatkuva kehittäminen, niin voit saavuttaa täydellisyyden.
10. **Jonoteoria**. Ymmärrä millä tavalla jonoja hyödyntävät tietojärjestelmät toimivat tuotekehityksessä. Sovella näitä oivalluksia hallitessasi jonojen kokoa, työnalla olevien tehtävien määrää, useiden tehtävien samanaikaista toteuttamista, tehtäväkokonaisuuksia ja tehtävien vaihtelevuutta.

LeSS yhdistää hyvin pitkälle Scrumin, Lean-filosofian ja ketterien menetelmien periaatteita, ja tarjoaa niiden skaalaamiseen suureen organisaatioon tai laajaan projektikokonaisuuteen tiettyjä toimenpiteitä. LeSS-viitekehys määrittelee lisäksi sääntöjä, jotka ohjaavat LeSS-menetelmien käyttämistä, samalla tavalla kuin Scrum esittää toimintaohjeita, joidan avulla kyseistä menetelmää hyödynnetään. LeSS:n toimintatavat on kuvattu viitekehyksen säännöissä (Larman & Vodde 2015). Kuten Scrum ja Kanban, myös LeSS on empiirinen menetelmä, joka edellyttää sen käyttäjältä kokeilua ja parhaiden toimintatapojen löytämistä menetelmää hyödyntävän organisaation yksilölliseen tilanteeseen.

3. TYÖN AINEISTO, MENETELMÄT JA AINEISTON ANALYYSI

Tässä luvussa esitetään diplomityön taustat ja lähtökohdat tutkimukselle. Luvussa käydään läpi tutkimukselle asetetut tavoitteet ja tutkimuskohteet. Tämän jälkeen esitetään valitut tutkimusmenetelmät ja kuvataan aineiston kokoamisen periaatteet. Luvun pääasiallisena sisältönä on toteuttaa työn kohteena olevan aineiston analyysi valittuja tutkimusmenetelmiä hyödyntäen.

3.1 Työn tausta

Tämä diplomityö on tehty syksyn 2015 aikana, työskennellessäni projektipäällikkönä helsinkiläisessä IT-alan yrityksessä. Yritys kehittää autoalan toimijoille moderneja tietojärjestelmiä, joiden tarkoituksena on yhdistää kaikki autoiluun ja ajoneuvoon liittyvät osapuolet kuten omistaja, kuljettaja, automyyjä, huoltokorjaamo, vakuutusyhtiö, tulli, verottaja sekä muut mahdolliset toimijat.

Diplomityön taustalla on toukokuussa 2014 alkanut kansainvälinen asiakasprojekti, jonka asiakkaana on suuri kansainvälinen pörssiyhtiö. Hankkeessa toteutetaan asiakkaalle laajamittainen tietojärjestelmäuudistus, jossa heidän nykyinen taloudenhallinnan ja toiminnanohjauksen tietojärjestelmänsä korvataan uudella ja nykyaikaisella tietojärjestelmäkokonaisuudella. Hankkeen toteutus tehdään usean IT-alan yrityksen muodostaman projektiryhmän toimesta. Hanke koostuu useasta pienemmästä osaprojektista, joissa järjestelmätoimittajat työskentelevät omien vastuualueidensa mukaisesti. Järjestelmätoimittajien vastuualueet vaihtelevat suuresti, sillä tietyt toimittajat osallistuvat vain yhden osaprojektin toteutukseen, kun toisilla useita tiimejä voi työskennellä eri osaprojektien parissa.

Hanke toteutetaan useassa vaiheessa ja osajärjestelmien julkaisuaikataulut ovat osittain toisistaan riippumattomia. Jokaista osajärjestelmää kehitetään iteratiivisella lähestymistavalla, laajentaen näin järjestelmien toimintoja inkrementaalisesti. Kokonaisuuden laajamittainen käyttöönotto etenee osaprojektien ja niiden julkaisujen mukaisesti. Suunnitellussa tietojärjestelmäkokonaisuudessa osaprojektien mukaiset tietojärjestelmät ovat tiiviissä yhteydessä toisiinsa. Vaiheistuksen ansiosta osajärjestelmät linkittyvät julkaisujen myötä toisiinsa, ja näin tietojärjestelmäkokonaisuus, joka täyttää asiakkaan liiketoiminnan tulevaisuuden tarpeet, rakentuu pala palalta. Hanke on vaiheistettu osaprojektien lisäksi myös kansainvälisellä tasolla. Lähtökohtaisesti hankkeen jokaisen osaprojektin toteutustyö aloitetaan Suomessa. Tietojärjestelmän kehitys ja käyttöönottaminen

laajentuu seuraavaksi Ruotsiin ja Norjaan. Tämän edellytyksenä on se, että Suomessa on ensin päästy tilanteeseen, jossa osajärjestelmät ja niiden muodostama kokonaisuus on asiakkaan määrittämässä laajuudessa otettu onnistuneesti käyttöön. Hankkeen tarkoituksena on toteuttaa asiakkaalle yhtenäinen, maariippumaton kokonaisuus, joka soveltuu eri maiden mahdollisesti toisistaan hieman poikkeaviin tarpeisiin ja toimintatapoihin.

Hankkeen tavoitteena on korvata asiakkaan käytössä oleva järjestelmä, joka on toiminnaltaan ja kehitykseltään vanhentunut. Nykyinen järjestelmä ei pysty vastaamaan asiakkaan liiketoiminnan kehittymiseen ja tulevaisuuden näkymien mukanaan tuomiin liiketoimintaympäristön uusiin tarpeisiin. Käynnistämällä hankkeen asiakas pyrkii varmistamaan kilpailukykynsä pitkällä tähtäimellä. Tämä oli tärkein syy sille, että hankkeessa päätettiin uusia kaikki yrityksen tietojärjestelmät samassa yhteydessä. Jatkamalla nykyisten tietojärjestelmiensä käyttämistä, asiakas olisi voinut jatkaa toimintaansa, mutta se olisi vienyt heiltä pohjan tulevaisuuden merkittäviltä jatkokehityshankkeilta, joita liiketoiminnan kehitys olisi tuonut mukanaan. Tällaisessa tilanteessa järjestelmäuudistuksen tekeminen myöhemmässä vaiheessa tarkoittaa hyvin usein suurempaa muutostyötä ja siten kasvavia kustannuksia.

Toimin projektipäällikkönä yrityksessä, joka toimii hankkeessa lähellä asiakkaan pääasiallista liiketoimintayksikköä, sillä yrityksemme kehittää asiakkaan huoltokorjaamoilla ja jälleenmyynnin toimipaikoilla näkyvän osan tietojärjestelmäkokonaisuudesta. Tässä toteutuksessa käytetään pohjana yrityksemme omaa, autoalan toimijoille kehittämää sovellusta, jota laajennetaan ja jatkokehitetään hankkeen myötä vastaamaan asiakkaan sovellukselle asettamia lisätarpeita. Näillä toiminnallisuuksilla mahdollistetaan toimipaikkojen työn ja prosessien tehostaminen. Tämän lisäksi yrityksemme toteuttama sovellus laajentaa asiakkaan nykyisiä kuluttajarajapintoja uusille ulottuvuuksille tuomalla heidän loppuasiakkailleen uusia, kehittyneitä, ja ennen kaikkea autoilijan arkea helpottavia palveluja, muun muassa mobiilisovellusten ja sähköisten palvelukanavien muodossa.

3.2 Tutkimusmenetelmät ja aineisto

Tässä luvussa esitetään tutkimuksen tavoitteet ja tutkimusongelma. Lisäksi esitellään tutkimusmenetelmät, jotka on valittu työn toteuttamiseen.

3.2.1 Tutkimuksen tavoitteet ja tutkimusongelman kuvaaminen

Tämän diplomityön tarkoituksena on kuvata, miten ketterät ohjelmistokehitysmenetelmät soveltuvat suureen tietojärjestelmähankkeeseen, joka koostuu useasta pienemmästä projektista. Näiden osaprojektien toteuttamisessa on mukana useita järjestelmätoimittajia ja muita osapuolia, jotka yhdessä muodostavat kokonaishankkeen projektiorganisaation. Työn pääasiallisena tavoitteena on vastata seuraavaan tutkimuskysymykseen: Mi-

ten ketterät ohjelmistokehitysmenetelmät soveltuvat usean järjestelmätoimittajan väliseen tietojärjestelmähankkeeseen? Tätä kysymystä varten työssä otetaan kantaa myös seuraaviin kysymyksiin: Mitä ketteriä ohjelmistokehitysmenetelmiä tutkimuksen kohteena olevassa tietojärjestelmähankkeessa käytetään? Millä tavalla näitä menetelmiä käytetään? Mitä ketterien menetelmien käytäntöjä hankkeessa hyödynnetään?

Tutkittava tietojärjestelmähanke on aloitettu toukokuussa 2014, mutta ketterät menetelmät otettiin hankkeessa käyttöön vasta tammikuusta 2015 alkaen. Suoritettava tutkimus rajataan pääosin diplomityön kirjoitushetken tilanteeseen, eli siihen, millaisia käytäntöjä ja menetelmiä hankkeessa hyödynnetään vuoden 2015 syksyllä. Tutkimuksen yhteydessä on kuitenkin syytä perehtyä niihin syihin ja merkityksiin, joiden vuoksi hankkeessa siirryttiin käyttämään ketteriä ohjelmistokehitysmenetelmiä. Jotta hankkeessa tällä hetkellä käytössä olevia menetelmiä voidaan arvioida, otetaan tutkimuksen yhteydessä lyhyesti kantaa myös seuraaviin kysymyksiin: Millä tavalla hanketta ohjattiin ja toteutettiin vuoden 2014 aikana? Mistä syystä ketterät menetelmät otettiin käyttöön? Millä tavalla toimintatapojen muuttaminen on vaikuttanut hankkeen etenemiseen?

Diplomityön tavoitteena on osoittaa tietojärjestelmähankkeen asiakkaalle se, toimitaanko projektissa tällä hetkellä parhaalla mahdollisella tavalla. Työn tuloksien on tarkoitus antaa konkreettisia ehdotuksia siitä, millä tavalla hankkeen toimintatapoja olisi kannattavaa muuttaa, jotta hanke etenee jatkossa entistä paremmin. Näiden tavoitteiden ohella, työn tulosten on tarkoitus kuvata millä tavalla meidän yrityksemme näkee hankkeen edistymisen tällä hetkellä. Mitä ongelmia ja haasteita usean järjestelmätoimittajan väliseen yhteistyöhön liittyy? Mitkä asiat ovat vastaavasti onnistuneet hyvin? Ennen kaikkea pyritään kuvaamaan missä asioissa voidaan parantaa ja tehostaa toimintaa. Näiden vastausten toivotaan antavan sekä hankkeen asiakkaalle että kirjoittajan omalle yritykselle näkökulmia siihen, kuinka molempien osapuolien toimintaa voitaisiin kehittää.

Tutkimuksessa ei kateta koko tietojärjestelmähankkeen kaikkia osaprojekteja ja toteutusvaiheita. Tässä työssä keskitytään kuvaamaan pääosin sitä toteutusvaihetta ja osakokonaisuutta, jossa yrityksemme on mukana. Tämä toteutusvaihe on kuitenkin hankkeen merkittävin kokonaisuus, ja siihen liittyy enemmän osapuolia kuin muihin osaprojekteihin. Tutkimukselle saadaan tämän ansiosta joka tapauksessa merkittävää kattavuutta, ja saavutettuja tuloksia voidaan hyödyntää myös hankkeen kokonaislaajuudessa. Tuloksista voidaan pyrkiä tunnistamaan hyviä käytäntöjä ja toimintapoja myös hankkeen muihin osaprojekteihin ja toteutusvaiheisiin. Tutkimuksessa voidaan tietyiltä osin viitata myös hankkeeseen kokonaisuutena, mikäli näkökulma on tutkittavien ketterien menetelmien periaatteiden kannalta mielekästä tuoda esille.

3.2.2 Valitut tutkimusmenetelmät, aineiston kokoaminen ja analysointi

Tämän diplomityön tutkimusmenetelmänä käytetään tapaus- eli case-tutkimusta. Tapaustutkimuksen tavoitteena on tyypillisesti ilmiöiden kuvailu. Tavoitteeseen päästään valitsemalla yksittäinen tapaus, tilanne tai joukko tapauksia, joissa tutkimuksen kohteena on yksilö, ryhmä tai yhteisö. Tapaustutkimuksessa prosesseja ja tapauksia tutkitaan yhteydessä niiden ympäristöön, ja aineistoa on mahdollista kerätä muun muassa kyselyillä, haastattelemalla tai havainnoimalla tapauksia. (Hirsjärvi et al. 2007)

Yinin (2003) mukaan tapaustutkimuksella on kolme mahdollista tutkimusmetodia: 1) selittävä, 2) tutkiva ja 3) kuvaileva. Metodit eivät ole toisiaan pois sulkevia, vaan niitä voidaan hyödyntää myös yhdessä. Tapaustutkimuksen tarkoituksena on:

- selittää syy-yhteydet tutkittaville asioille
- kuvailla tutkimuskohteen kehittymistä
- havainnollistaa tiettyjä aiheita kehittymisessä
- tutkia tilanteita, joissa havainnoilla ei ole ainoastaan yhtä lopputulosta sekä
- meta-arviointi eli tutkimusarvioiden tutkiminen.

Schrammin (1971) mukaan jokaiselle tapaustutkimuksen tutkimusmetodille on ominaista pyrkiä valaisemaan ratkaisua tai ratkaisuja. Tutkimuksella vastataan kysymyksiin miksi ratkaisut valittiin, miten niitä käytettiin ja millaisin tuloksin. (katso Yin 2003) Yinin (2003) mukaan tapaustutkimusta suositellaan valittavaksi tutkimusmenetelmäksi, jos seuraavat kohdat tutkimuskohteessa täyttyvät:

- tutkimuskysymys vastaa kysymykseen “miten” tai “miksi”
- tutkijalla on vain vähän kontrollia tutkittaviin tapauksiin ja
- kun tapauksissa keskitytään tosielämän tilanteissa esiintyviin ilmiöihin.

Tutkimuksessa käytetään kvalitatiivista eli laadullista lähestymistapaa. Kvalitatiivisen lähestymistavan lähtökohtana on todellisen elämän kuvaaminen mahdollisimman kokonaisvaltaisesti. Laadullisessa tutkimuksessa aineiston keräämisessä suositaan tutkijan omia havaintoja ja niihin luotetaan enemmän kuin mittausvälineisiin. (Hirsjärvi et al. 2007) Laadullisen lähestymistavan myötä työn tutkimusmenetelmäksi muodostuu deskriptiivinen tapaustutkimus, jolla pyritään ymmärtämään käsiteltävän kokonaisuuden toiminnan merkitystä. Työn tavoitteena on muodostaa syvällisempi käsitys kokonaisuudesta työn aihepiirin yhteydessä.

Tutkimuksen aineiston kokoamisessa käytetään osallistuvaa havainnointia, jolle on tyypillistä se, että tutkija osallistuu havainnoitavan kohteen toimintaan. Kyselyllä ja haastattelulla voidaan tutkimusmenetelminä selvittää mitä henkilöt ajattelevat, tuntevat ja uskovat. Havainnointi sen sijaan kertoo, mitä tutkittavat todella tekevät, ja toimivatko

he todella niin kuin sanovat toimivansa. Havainnoinnin merkittävin etu on se, että tutkittavasta kohteen toiminnasta ja käyttäytymisestä saadaan välitöntä tietoa. Tästä syystä osallistuva havainnointi sopii erityisen hyvin kvalitatiivisen lähestymistavan tutkimusmenetelmäksi. (Hirsjärvi et al. 2007)

Tutkimuksen aineisto on muodostunut kirjoittajan empiirisen, kokemusperäisen havainnoinnin avulla. Kirjoittaja on ollut toimenkuvansa johdosta erittäin tiiviisti osallisena havainnoitavassa kokonaisuudessa. Tapaustutkimuksen aineisto on kerätty hankkeen aloituksesta alkaen vajaan puolentoista vuoden aikana tämän diplomityön kirjoitushetkeen asti, marraskuulle 2015. Tänä aikana kirjoittaja on osallistunut yli 200 tapaamiseen, joiden joukossa on sekä laajoja hankkeen ohjausryhmätapaamisia, yksittäisiä osajärjestelmäkohtaisia määrittely- ja suunnittelutapaamisia, sekä päivittäisiä statuspalavereja. On huomioitava, että tietojärjestelmähanke on edelleen käynnissä tämän diplomityön kirjoitushetkellä, joten tutkimuskohteen havainnointi jatkuu myös tämän tutkimuksen tulosten esittämisen jälkeen. Hankkeessa on kuitenkin käytössä vakiintuneet toimintatavat ja menetelmät, joten deskriptiivisen tapaustutkimuksen suorittaminen on mahdollista rajaamalla aineisto tutkimuksen suorittamishetkeen.

Scapensin (1990) mukaan tapaustutkimuksen suorittamiseen liittyy kolme yleisesti tunnistettua haastetta tai rajoitetta. Ensimmäinen on tapaustutkimuksen heikko yleistettävyys ja akateeminen epätäsmällisyys. Onnistuneessa tapaustutkimuksessa tehtyjä havaintoja selitetään tunnistettuja teorioita käyttämällä. Toinen haaste on tapaustutkimuksen kattavuus. Tällä viitataan siihen, että tutkijan on yleensä mahdotonta ottaa huomioon kaikkia tutkimukseen liittyvä näkökulmia. Tapaustutkimuksen suorittamisessa on erityisen tärkeää tehdä selkeä rajaus tutkimuksen kohteesta. Kolmantena haasteena voidaan pitää tutkijan objektiivisuuden säilyttämistä. Tapaustutkimusta suorittaessaan tutkijan on mahdotonta toimia täysin neutraalisti, kun tulkitaan sosiaalista todellisuutta, johon tutkija ottaa itse osaa. Tutkijan havainnoinnista johtuvaa subjektiivista vääristymää pyritään vähentämään käyttämällä useampaa menetelmää tiedon keräämisessä sekä dokumentoimalla havainnot huolellisesti. (katso Hakansuu 2011)

Hirsjärvi et al. (2007) mukaan tutkimuksella on aina jokin tarkoitus ja tehtävä. Tutkimusstrategian valintaa ohjaa tämä tarkoitus. Hirsjärvi et al. (2007) esittävät Yinin (2003) kuvaamat kolme kysymyssarjaa, joiden avulla löydetään tilanteeseen sopiva tutkimusstrategia:

1. Onko tutkimusongelma luonteeltaan kartoittava? Pyrkiikö se kuvaamaan tapahtumaa tai ilmiötä? Vai yrittääkö se selittää sosiaalista ilmiötä?
2. Vaatiiko tutkimus suorittajalta toimintojen kontrolloimista? Vai pyrkiikö tutkimus kuvaamaan luonnollisesti tapahtuvia ilmiöitä?
3. Sijoittuuko tutkimuksen kohde luonteeltaan nykyhetkeen vai menneisyyteen?

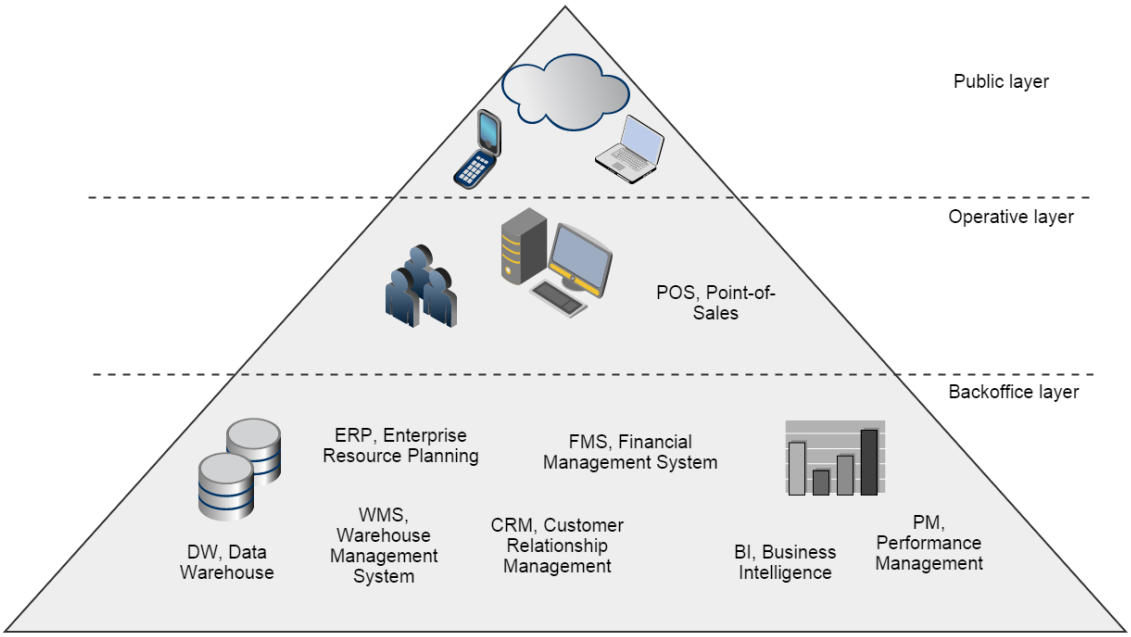
Vastaamalla tämän työn tavoitteiden ja tutkimusongelman osalta tässä esitettyihin kysymyksiin, voidaan todeta deskriptiivisen tapaustutkimuksen olevan hyvä valinta tässä diplomityössä käsiteltävän tutkimuksen suorittamiseen. Valituilla menetelmillä ja lähestymistavoilla voidaan saavuttaa selkeitä tuloksia diplomityön osalta.

3.3 Case: ketterät menetelmät usean järjestelmätoimittajan IT-hankkeessa

Tässä luvussa suoritetaan kattava tapaustutkimus työn kohteena olevasta tietojärjestelmähankkeesta.

3.3.1 Tietojärjestelmähankkeen yleiskuvaus

Hankkeessa toteutetaan autoalan toimijalle kokonaisvaltainen tietojärjestelmäuudistus, johon osallistuu useita järjestelmätoimittajia. Selkeyden vuoksi asiakkaasta käytetään tässä tapaustutkimuksen yhteydessä termiä “asiakasyritys”. Tavoitteena on toteuttaa asiakasyrityksen nykyisiä tarpeita vastaava kokonaisuus, joka huomioi mahdollisimman tehokkaasti ja kattavasti asiakasyrityksen tulevaisuuden liiketoimintatarpeet. Tulevaisuuden tarpeiden huomioiminen koostuu kahdesta osasta: 1) tunnistaa jo tiedossa olevat kehitystarpeet ja toiminnallisuudet sekä 2) toteuttaa joustava kokonaisratkaisu, joka mahdollistaa tulevaisuuden uusien vaatimusten täyttämisen skaalautuvan ja muokattavan järjestelmätoteutuksen ansiosta.



Kuva 7. *Hankkeen kolme tietojärjestelmätasoa.*

Projekti koostuu useasta osajärjestelmästä, jotka yhdessä muodostavat asiakasyrityksen vaatimukset ja tavoitteet täyttävän tietojärjestelmäkokonaisuuden. Kuva 7 esittää koko-

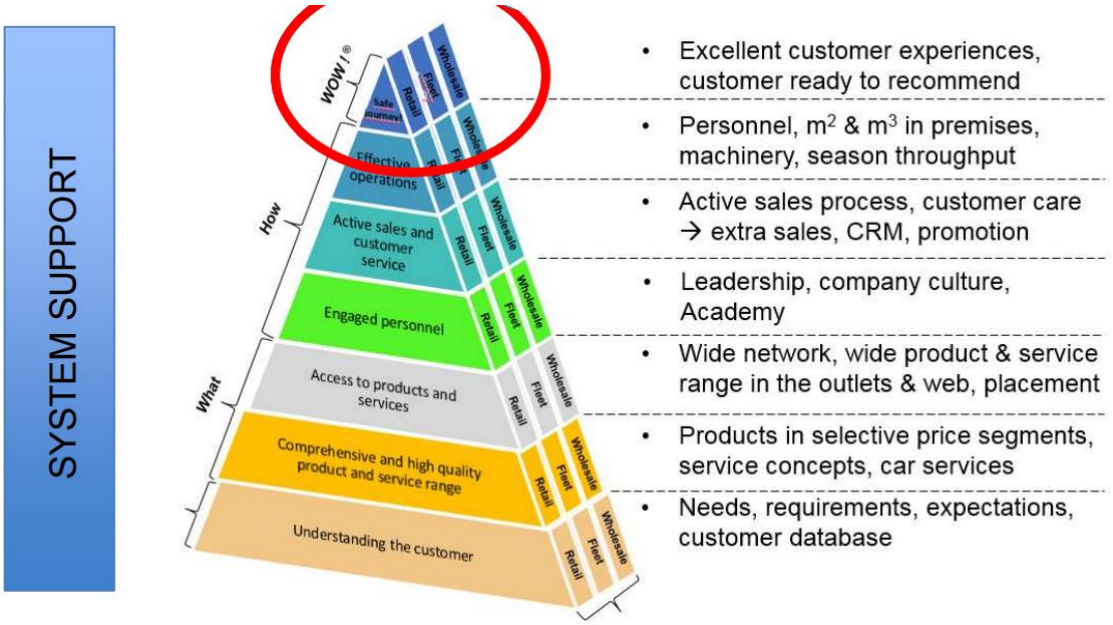
naisratkaisun rakenteen kolmiportaisena pyramidina. Alimmaisena on niin sanottu *"Backoffice"* -taso, joka kuvaa sitä osaa tietojärjestelmäkokonaisuudesta, jolla asiakasyrityksen liiketoiminta ei tee sille varsinaista rahallista tulosta, ainakaan merkittävissä määrin. Tämä taso on kuitenkin pakollinen osa kokonaisuutta, jotta kuvassa sen yläpuolella olevat kolmion seuraavat tasot voivat toimia. Käytännön tasolla ajateltuna, alimpana olevaan tasoon kuuluvat yrityksen taloudenhallinnan, raportoinnin, varaston- ja toimitusketjujenhallinnan, käyttäjä- ja asiakashallinnan tietojärjestelmät sekä tietojärjestelmäkokonaisuuden keskitetyt tietovarannot.

Pyramidin keskimmäinen taso, *"Operative layer"*, kuvaa asiakasyrityksen pääasiallista liiketoimintaympäristöä. Käytännössä tämä taso sisältää kaikki ne osajärjestelmät, joita asiakasyrityksen operatiiviset liiketoimintayksiköt käyttävät. Tällä tasolla olevat osajärjestelmät muodostavat asiakasyrityksen liiketoiminnan tuloksesta merkittävimmän osan. Operatiiviset tietojärjestelmät eivät yksinään toimi, vaan ne tarvitsevat pohjakseen *"Backoffice"*-tason osajärjestelmiä. Vastaavasti pyramidin huipulla oleva *"Public"* -taso, vaatii toimiakseen operatiivisen tason tietojärjestelmiä. Ylin taso kuvaa julkisen kerroksen tietojärjestelmiä, jotka on tarkoitettu asiakasyrityksen omien loppuasiakkaiden käyttöön. Näitä järjestelmiä ovat esimerkiksi verkkokaupat, mobiilisovellukset ja muut sähköiset palvelut, joita asiakasyritys voi tarjota loppuasiakkailleen, niin yksityishenkilöille kuin isommille yritysasiakkaille.

Pyramidin eri tasojen pinta-alat havainnollistavat tietojärjestelmien kokoluokkaa. Alimmalla tasolla olevat osajärjestelmät ovat kokoluokaltaan projektin suurimmat tietojärjestelmät. Liiketoiminnan kehityksen ja yrityksen tuloksen näkökulmasta tietojärjestelmien perusta on erityisen tärkeää, vaikka asiakasyrityksen euromääräinen tulos tehdään ylemmillä tasoilla olevilla tietojärjestelmillä niiden pienemmästä koosta huolimatta.

3.3.2 Asiakasyrityksen liiketoiminnan arvot ja tavoitteet projektille

Asiakasyritys on kuvannut liiketoimintansa arvot kuvan 8 mukaisella arvopyramidilla. Tämä noudattaa samanlaista pyramidirakennetta, jolla esitettiin edellä järjestelmätasojen väliset suhteet. Arvopyramidin eri tasot kuvaavat asiakasyrityksen liiketoiminnan arvoja ja tavoitetilaa, jossa asiakasyritys haluaa toimia jokaisen liiketoimintayksikkönsä kohdalla. Kuten edellisessä kuvassa, myös tässä, pyramidin pohja on perustana ylempien tasojen toteutumiselle ja saavuttamiselle, aina pyramidin huipulle asti. Pyramidin huippu on se tavoite, jota kohti asiakasyritys haluaa toiminnallaan pyrkiä. Kuvassa on tuotu esille asiakasyrityksen toiminnan laajuus kohdentamalla arvot erikseen jokaiselle asiakassegmentille. Asiakassegmentit kuvataan pyramidissa z-akselilla ja ne ovat asiakasyrityksen osalta vähittäiskauppa (*"Retail"*), suurasiakkuudet (*"Fleet"*) ja tukkukauppa (*"Wholesale"*).



Kuva 8. *Asiakasyrityksen liiketoiminnan arvot ja projektin tavoitteet (Asiakasyrityksen esitys hankkeen välistatustapaamisessa 2014).*

Asiakasyrityksen liiketoiminnan arvot koostuvat seuraavista tasoista, alkaen pyramidin pohjalta:

1. **Asiakkaan tarpeiden ymmärtäminen**. Tässä listauksessa asiakkaalla tarkoitetaan asiakasyrityksen loppuasiakkaita, jotka käyttävät asiakasyrityksen tarjoamia palveluja. Asiakkaan tarpeiden ymmärtäminen on perusta koko liiketoiminnalle ja sen kehittämiselle.
2. **Kattava ja korkealaatuinen tuote- sekä palveluvalikoima**. Ymmärtämällä asiakkaidensa tarpeet asiakasyritys pystyy vastaamaan näihin tarjoamalla kattavaa ja korkealaatuista tuote- ja palveluvalikoimaa, joka tyydyttää asiakkaiden tarpeet.
3. **Tuotteiden ja palveluiden saatavuus**. Tuote- ja palveluvalikoima täytyy tuoda saataville, jotta asiakkailla on mahdollisuus hyödyntää asiakasyrityksen palveluja ja hankkia sen tarjoamia tuotteita käyttöönsä.
4. **Sitoutunut henkilöstö**. Henkilöstön avulla tuotteiden ja palveluiden saatavuudesta tulee konkreettista, jolloin asiakkaat voivat asioida asiantuntevan henkilöstön kanssa.
5. **Aktiivinen myynti ja asiakaspalvelu**. Asiakkaan palvelemisella ja aktiivisella myynnillä hahmotetaan yksilölliset asiakastarpeet ja löydetään niihin parhaiten soveltuvat ratkaisut asiakasyrityksen tuotteiden ja palveluiden muodossa.
6. **Tehokkaat toiminnot**. Liiketoiminnan tehokkailla toiminnoilla ja välineillä mahdollistetaan saumaton ja tehokas asiakastarpeisiin vastaaminen.
7. **Erinomainen asiakastyytyväisyys**. Vastaamalla tunnistettuihin asiakastarpeisiin oikeilla tuotteilla ja palveluilla, sekä niiden tehokkaalla ja asiantunte-

> valla toimittamisella, asiakasyritys voi saavuttaa erinomaisen asiakastyytyväisyyden ja tilanteen, jossa asiakkaat ovat valmiita suosittelemaan saamiaan palveluita potentiaalisille uusille asiakkaille. Tämä on se tilanne, jota asiakasyritys tavoittelee keskittymällä liiketoimintansa arvojen toteuttamiseen.

Arvopyramidi ohjaa myös tässä hankkeessa toteutettavan tietojärjestelmäkokonaisuuden toteutusta. Pyramidi osoittaa kuinka toteutettavalla kokonaisuudella täytetään asiakasyrityksen liiketoiminnalle asetetut arvovaatimukset. Kuvassa 8 tätä esittää vasemmalla oleva sininen palkki, joka kuvaa tietojärjestelmiä liiketoiminnan mahdollistajana (*"System support"*). Ensimmäinen taso, asiakasymmärrys, määrittelee vaatimukset jokaiselle osatietojärjestelmälle. Kaikkien osajärjestelmien toteutuksessa on ymmärrettävä asiakasyrityksen lähtökohtaiset tarpeet, jotta osajärjestelmät voivat täyttää ja vastata asiakasyrityksen nykyisiin ja tulevaisuuden uusiin liiketoimintavaatimuksiin.

Seuraavat kaksi tasoa - tuote- ja palveluvalikoima sekä näiden saatavuus - antavat tietojärjestelmäkokonaisuudelle yleisen vaatimuksen siitä, mitä sen täytyy sisältää (*"What"*). Asiakasyrityksen kannalta heidän liiketoiminnan mahdollistamisen perustana on pystyä tarjoamaan heidän loppuasiakkailleen asiakasyrityksen tuotteita ja palveluita. Tietojärjestelmäkokonaisuuden jokaisen järjestelmätason on mahdollistettava tämä. Arvopyramidin tasot 4-6 vastaavat kysymykseen miten asiakasyrityksen toiminnan tehokkuus ja liiketoiminnalliset tulokset saavutetaan (*"How"*). Toteutettavien osajärjestelmien tulee tukea tehokasta toimintaa ja loppuasiakkaan jatkuvaa huomioimista, jotta liiketoiminnan arvot ja arvopyramidin huippu on mahdollista saavuttaa. Huipulla pyritään tuottamaan loppuasiakkaalle elämys ja ennennäkemätön tyytyväisyys asiakasyrityksen toimintaan. Tämä esitetään kuvassa *"WOW!"* -efektillä.

Tässä kohtaa on syytä huomioida edellä esitettyjen kahden pyramidin välinen yhteys. Arvopyramidin alimmainen taso asettaa vaatimukset tietojärjestelmäkokonaisuuden kaikille tasoille. Sen sijaan arvopyramidin ylimmäinen *"WOW!"* -efekti linkittyy vain operatiivisen liiketoiminnan *"Operative"* ja *"Public"* tietojärjestelmätasoille, ja näistä vielä selkeästi enemmän *"Public"*-tason palveluiden merkitystä korostaen. Näiden arvopyramidin alimman ja ylimmän tason välissä olevat tasot, koskettavat jokaista tietojärjestelmätasoa. Myös näiden tasojen osalta, tasojen välinen suhde korostuu kohti pyramidien ylempiä tasoja. Tämä tarkoittaa sitä, että arvopyramidin *"What"* -taso kohdentuu enemmän *"Backoffice"*-tasolle kuin *"Public"*-tasolle, mutta sitä ei kuitenkaan voida jättää täysin huomioimatta ylimmälläkään tietojärjestelmätasolla. Arvopyramidin *"How"*-tasojen merkitys vastaavasti korostuu selkeästi enemmän tietojärjestelmän operatiivisilla kuin *"Backoffice"*-tasolla.

Edellä esitetyistä huomioista voidaan nähdä yhteys sille, että arvojen määrittelemä *"Mitä"* korostuu tietojärjestelmätason pohjalla, ja vastaavasti *"Miten"* korostuu liiketoiminnan operatiivisen toiminnan tietojärjestelmätasoilla. Tässä välissä on tarpeen myös huomauttaa, että tällä hetkellä operatiivinen taso tuottaa asiakasyrityksen liiketoiminnan

tuloksesta suurimman osan. Asiakasyrityksen visiona ja pitkän aikavälin tavoitteena on kuitenkin se, että julkisen tason tietojärjestelmät tuottavat pitkällä aikavälillä merkittävimmän tuloksen asiakasyritykselle, tarjoamalla sen loppuasiakkaille todellista *"WOW!"* -efektiä. Asiakasyrityksen mukaan loppuasiakkaiden todellinen tyytyväisyys on perustana liiketoiminnan kehitykselle ja menestymiselle tulevaisuudessa.

3.3.3 Projektiorganisaatio, osapuolet ja vastuualueet

Tietojärjestelmähankkeen projektiorganisaatio koostuu asiakasyrityksen ja järjestelmätoimittajien edustajista. Asiakasyritys toimii hankkeen tilaajana. Asiakasyrityksen vastuulla on hankekokonaisuuden budjetointi, koordinointi, projektiohjaus, aikataulut, sekä sisällön ja vaatimusten esittäminen. Asiakasyrityksen sisällä hankevastuu mainituista asioista on jaettu osittain osaprojekteihin, ja näiden vastuuhenkilöille. Esimerkiksi osaprojektien toisistaan poikkeavat aikataulut ja sisältövaatimukset voidaan päättää itsenäisesti osaprojektin vastuuhenkilön tai -henkilöiden toimesta. Jokaisella osaprojektilla on asiakasyrityksessä *"Interest group" (sidosryhmä)*, jolle osaprojektin toteutuksen edistyminen raportoidaan vastuuhenkilöiden toimesta.

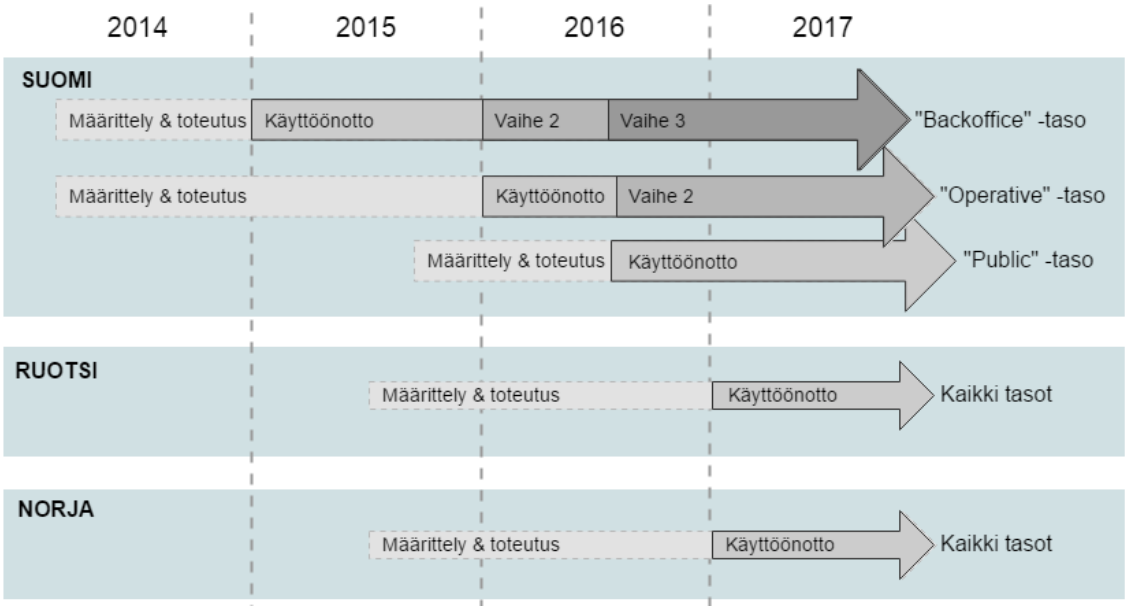
Järjestelmätoimittajat vastaavat omalta osaltaan heitä koskevien osaprojektien toteutustyöstä. Järjestelmätoimittajien tehtävänä on käydä yhdessä asiakasyrityksen edustajien kanssa asiakastarpeet ja liiketoimintavaatimukset läpi. Järjestelmätoimittajien vastuulla on löytää parhaat mahdolliset ratkaisut vaatimusten ja asiakastarpeiden toteuttamiseen määrittelyiden ja suunnittelutapaamisten avulla. Osaprojektiensa toteuttamisen ohella toimittajat vastaavat eri tietojärjestelmien sitomisesta toisiinsa, jos tiettyjen tietojärjestelmien välillä täytyy välittää informaatiota. Osaprojektien toteutuksen aikataulutuksen järjestelmätoimittajat tekevät yhteistyössä asikasyrityksen kanssa. Tietojärjestelmien käyttöönottamiseen liittyvät toiminnot ja käyttöä seuraavat tukitoiminnot kuuluvat jokaisen järjestelmätoimittajan vastuulle.

Esitettyjen osapuolien lisäksi hankkeeseen voi kuulua muita sidosryhmiä, lähinnä asiakasyrityksen sisäisestä organisaatiorakenteesta. Näitä ei ole tarkoituksenmukaista esitellä tarkemmin tämän case-tutkimuksen yhteydessä, sillä nämä sidosryhmät eivät vaikuta tutkimuskohteen havainnointiin.

3.3.4 Hankkeen aikataulu

Hankkeen aikataulu on esitetty yleisellä tasolla kuvassa 9. Tietojärjestelmien käyttöönotto on vaiheistettu maakohtaisesti. Ensimmäisenä toteutetaan ja käyttöönotetaan jokainen järjestelmätaso Suomessa. Tämän jälkeen Suomessa toteutettua kokonaisuutta täydennetään Ruotsin ja Norjan asettamilla jatkovaatimuksilla, joiden toteuttamisen jälkeen näiden maiden käyttöönotto voidaan aloittaa. Pohjoismaiden jälkeen hanke laajentuu asiakasyrityksen muihin liiketoimintamaihin, mutta näiden osalta aikataulu ei ole vielä tiedossa. Kuvassa esitetyt eri vaiheiden käyttöönottamisen aikataulut ovat työn kirjoi-

tushetkellä tiedossa olevan vaiheistuksen mukaisia. Osaprojektien toteutuksen edetessä nämä voivat muuttua tässä esitetystä ketterien menetelmien mukaisista sen suhteen, missä vaiheessa käyttöönottamisen vaatimat toiminnallisuudet on todettu julkaisuvalmiiksi.



Kuva 9. *Hankkeen aikataulu ja tietojärjestelmätasojen vaiheistus.*

Suomen osalta toteutustyö on jaettu kuvan osoittamalla tavalla tietojärjestelmätasojen mukaisesti vaiheisiin. Toteutettava tietojärjestelmäkokonaisuus valmistuu näiden vaiheiden mukaisesti inkrementaalisesti. Esitettyjen vaiheiden sisällä järjestelmätasojen toteuttaminen ja käyttöönotto voi jakautua vielä useampaan osaan, järjestelmätasojen sisällä olevien osajärjestelmien mukaan. Osajärjestelmien käyttöönottaminen voidaan tehdä joko pilotoimalla tai aloittamalla laajamittainen käyttö samalla hetkellä, riippuen kunkin osajärjestelmän käyttötarkoituksesta, kokoluokasta ja käyttäjämäärästä.

3.3.5 Ketterien menetelmien käyttöönottaminen

Ketterät ohjelmistokehitysmenetelmät otettiin käyttöön tietojärjestelmähankkeessa vuoden 2015 alussa. Ennen tätä hanke eteni perinteisiä menetelmiä mukaillen: Tietojärjestelmäkokonaisuutta määriteltiin asiakasyrityksen ja järjestelmätoimittajien toimesta useista eri näkökulmista. Määrittelyissä pyrittiin käymään mahdollisimman tarkasti läpi hankkeen jokaisen osaprojektin sisällöt ja vaatimukset. Järjestelmien toimintaprosesseja suunniteltiin yksittäisten toimintojen tasolle. Tämä tehtiin usein hyvin suuren osanottajoukolla, jossa oli asiakasyrityksen lisäksi mukana suurin osa järjestelmätoimittajista.

Yrityksemme rooli vuoden 2015 aikana toteutuksen keskiössä olevassa operatiivisen tason tietojärjestelmäkokonaisuudessa on järjestelmätoimittajista kaikista merkittävin. Kokonaishankkeen toteuttamisen kannalta tämä tarkoittaa sitä, että tämän vuoden aikana operatiivisen tason tietojärjestelmien vaatimukset ohjaavat myös muiden järjestelmä-

tasojen kehitystyötä. Ketterät menetelmät otettiin hankkeessa käyttöön yrityksemme aloitteesta. Tärkeimpänä tarpeena toimintatapojen muutokselle oli mielestämme se, että siirtymällä ketterien menetelmien käyttöön, pystyimme paremmin varmistamaan hankkeen toteutustyön tehokkaamman etenemisen.

Scrumin mukaisesti tietojärjestelmäkokonaisuus on kasvanut vuoden 2015 aikana inkrementaalisesti kohti niitä toiminnallisuuksia, joita operatiivisen tason käyttöönottovaiheessa 2016 vuoden alussa tarvitaan. Hankkeelta puuttui aiemmin selkeä koordinointi, jolla toteutustyötä olisi ohjattu ja vaiheistettu osajärjestelmien tasolla. Yrityksemme näkökulmasta toteuttamamme osajärjestelmän laajuus ja käyttöönottamisen aikataulu ovat olleet niin haastavat, että projektin toimintatapoja oli muutettava, jotta eri hankkeen vaiheiden valmistumista voidaan arvioida ja ohjata paremmin. Tämän tapaustutkimuksen seuraavissa luvuissa kuvataan tarkemmin kuinka ketteriä menetelmiä hankkeessa hyödynnetään.

3.3.6 Kokonaishankkeen projektinhallinta

Kokonaishankkeen edistymisestä vastaavat erilaiset ohjausryhmät ja kokoonpanot, joiden vastuualueet on jaettu asiakasyrityksen toimesta projektin kannalta loogisiin osioihin. Seuraavassa esitellään nämä ohjausryhmät, niihin osallistuvat tahot, sekä niiden toimenkuva ja vastuut.

"Steering Group" on kokonaishankkeen ohjausryhmä asiakasyrityksen puolella. Tämä ohjausryhmä vastaa hankkeen ylimmän tason päätöksistä kuten budjetoinnista ja hankkeen resurssoinnista asiakasyrityksen osalta. Ohjausryhmä antaa myös hankkeen aikataululle ohjeistuksen ja esittää näkökantansa hankkeen eri vaiheiden priorisoinnista. Ohjausryhmä pitää säännöllisiä tapaamisia noin kahden kuukauden välein. Näissä tapaamisissa käsitellään ohjausryhmän vastuulla olevia asioita.

"PMO" (Project Management Office) on asiakasyrityksen sisäinen status kokonaishankkeen eri osaprojektien osalta. PMO tekee hankkeeseen liittyvät operatiiviset päätökset. Näihin päätöksiin kuuluvat muun muassa aikataulujen kiinnitykset, eri osajärjestelmien toteutuksiin liittyvät vaiheistukset kuten tietyn osajärjestelmän testaus- tai tuotantokäytön aloittaminen. PMO siis päättää milloin kukin osajärjestelmä on valmis tuotantoon. PMO:n vastuulla on myös osajärjestelmien tuotantokäytön suunnitelmien sekä laadunvarmistusprosessien käytäntöönpano. PMO määrittelee ohjeistukset ja yleiset toimintatavat järjestelmätoimittajille hankkeen toteutustyöhön ja seurattavuuteen liittyen. PMO pitää säännöllisiä tapaamisia noin kuukauden välein. Näissä tapaamisissa käsitellään hankkeen osaprojektien tilannetta, kohdennetaan tarvittaessa lisäresursseja osaprojektien toteutukseen asiakasyrityksen osalta, sekä tehdään tarvittavat päätökset seuraavista toimenpiteistä osaprojektien tilanteiden mukaisesti. PMO:n tapaamisessa ovat pääosin läsnä vain asiakasyrityksen edustajia jokaisen osajärjestelmän ja toteutus-

vaiheen osalta, mutta myös järjestelmätoimittajien osallistuminen on tarvittaessa mahdollista.

"Group Project Team" on osajärjestelmien toteutustyötä seuraava ohjausryhmä, jonka jäsenissä ovat edustettuina asiakasyrityksen jokaisen osajärjestelmän projektipäällikkö tai vastuuhenkilö sekä jokaisen järjestelmätoimittajan projektipäälliköt. Tämän ohjausryhmän tarkoituksena on tuoda hankkeen jokaiselle osapuolelle tietoon kunkin osajärjestelmän ja toteutusvaiheen toteutuksen ajantasainen tilannekatsaus. Ohjausryhmän tapaamisessa jokaisen järjestelmätoimittajan projektipäällikkö käy vuorollaan läpi heitä koskettavien osajärjestelmien tilanteen. Projektipäällikköjen vastuulla on valmistella esityksensä ennen tapaamista. Asiakasyrityksen vastuuhenkilöt täydentävät tilannekatsausta tarvittaessa omilla näkemyksillään osajärjestelmien toteutuksen edistymisestä.

"Group Project Team" -tapaamiset muistuttavat Scrumin mukaisia *"Daily Scrum"* -tapaamisia. Jokaisen osajärjestelmän tilanteessa käydään läpi millä tavalla osajärjestelmän toteutus on edistynyt edellisen tapaamisen jälkeen, ja mitkä tuolloin asetetut tavoitteet on saavutettu ja onko jokin asetettu tavoite jäänyt saavuttamatta. Lisäksi käydään läpi mitä on tällä hetkellä toteutuksessa, ja mitä tavoitteita ja tehtäviä on vuorossa seuraavaksi. Projektipäälliköiden vastuulla on tuoda ilmi tunnistetut mahdolliset ongelmakohdat ja haasteet, jotka voivat vaikuttaa osaprojektin toteutukseen tällä hetkellä. Haasteet on jaoteltu tilannekatsauksen osalta seuraaviin kategorioihin: 1) aikataulu, 2) sisältö ja 3) resurssit. Kategorioihin lajitelluista haasteista ja ongelmista on lisäksi koostettu riskilistaus, joka tuo ilmi näiden kriittisyyden ja mahdollisen vaikutuksen osaprojektin aikatauluihin ja toteutukseen liittyen. Tapaamisen päätavoitteena on vastata kunkin osaprojektin osalta seuraaviin kysymyksiin: Miten osaprojektin toteutustyö etenee? Onko osaprojektille määritetyt tavoitteet valmistumassa niille asetettuun seuraavaan aikataulussa määritettyyn määräpäivään mennessä?

"Local Project Meetings" (paikalliset projektitapaamiset) ovat maakohtaisesti järjestettäviä tapaamisia, joita voivat olla esimerkiksi Sprint-suunnittelutapaamiset, viikoittaiset statuspalaverit, päivittäiset "Daily Scrum" -palaverit, sekä kaikki määrittely- ja suunnittelupalaverit, joita hankkeen toteuttamisessa tarvitaan. Nämä tapaamiset ohjaavat hankkeen tietojärjestelmien käytännön toteutustyötä jokaisen osajärjestelmän osalta, hankkeen vaiheistukset huomioiden. Paikallisten projektitapaamisten sisältö, käytännöt ja toimintatavat voidaan sopia maakohtaisesti parhaaksi katsotuilla tavoilla. Luvussa *3.3.11 Tapaamiset ja toteutuksen seuranta* käydään tarkemmin läpi, millaiset käytännöt paikallisissa projektitapaamisissa on tutkimuskohteen osalta ollut käytössä Suomessa.

3.3.7 Käytettävät työkalut ja niiden käyttötarkoitus

Projektissa käytetään useita tietoteknisiä työkaluja suunnitteluun, dokumentointiin, tehtävienhallintaan sekä projektiosapuolien väliseen kommunikointiin. Seuraavassa tehdään katsaus näihin työkaluihin kuvaamalla niiden käyttötarkoitukset.

JIRA Software (2015), jatkossa "Jira", on Australialaisen Atlassianin kehittämä tehtävienhallintaohjelmisto. Jiraa käytetään projektinhallinnan työkaluna hankkeen kehitystyön seurantaan. Jirassa tehdään Scrumin periaatteiden mukaisesti sprint-suunnitelmat, kehityksen status käydään viikoittain läpi Jiran Kanban-näkymien avulla, ja yksittäisten tehtävien seurantaa hallitaan Jiran tikettien avulla. Luvuissa *3.3.9 Projektinhallinta, toteutustyön aikatauluttaminen* sekä *3.3.11 Tapaamiset ja toteutuksen seuranta* kuvataan Jiran käyttötavat osaprojektissa, jossa yrityksemme toimii.

Confluence (2015) on Atlassianin kehittämä wikiohjelmisto, jota käytetään projektin dokumentaation keskitettyyn tuottamiseen, hallintaan ja jakamiseen. Kaikki wikiin tuotettu sisältö on jokaisen projektiosapuolen saatavilla. Dokumentoinnin käytännöt käydään tarkemmin läpi luvussa *3.3.12 Dokumentointi*.

Extranet on projektin keskitettynä dokumentinhallintana käytettävä Microsoftin kehittämä SharePoint-järjestelmä (2015). Extranetissa on projektille oma sivustonsa, joka on jaettu osaprojektikohtaisiin alisivuihin. Näille sivuille jokainen järjestelmätoimittaja ja asiakasyritys voivat tallentaa osaprojekteihin liityviä tai yleisiä kokonaisprojektia koskettavia dokumentteja ja muita tiedostoja. Sivuston kautta kaikki projektin jaettavat dokumentit ja tiedostot ovat jokaisen osapuolen saatavilla ja luettavissa, ellei dokumenttia ole erikseen rajattu vain tietylle joukolle. Extranet on ollut käytössä projektin aloituksesta alkaen, mutta sen rooli on pienentynyt Confluencen käyttöönoton myötä. Confluence mahdollistaa SharePointia paremmin dokumentaation tuottamisen, ja tähän liittyvien dokumenttien ja tiedostojen tallentamisen.

Flowdock (2015) on projektin pikaviestintään käytetty Rally Softwaren kehittämä sovellus, joka toimii useilla eri päätelaitteilla. Flowdock mahdollistaa tiimien välisen ryhmäkeskustelun ja yksityisviestien lähettämisen. Flowdockia käytetään ensisijaisena tiedonvälityskanavana kokonaisprojektin toteutustyön ja seurannan yhteydessä. Sovellukseen on luotu osajärjestelmä- ja vaihekohtaisia huoneita, joissa eri osapuolet voivat olla läsnä osallistuakseen keskusteluun. Flowdockin käyttämistä osana Scrumin mukaisia *"Daily Scrum"* -palavereja käsitellään tarkemmin luvussa *3.3.11 Tapaamiset ja toteutuksen seuranta*.

Sähköposti on käytössä projektin yleiseen viestintään. Sähköpostin käyttöä on pyritty korvaamaan Flowdockilla, koska sillä keskustelut pysyvät sähköpostia paremmin hallinnassa, tallessa ja lisäksi keskusteluihin on helpompi ottaa myös ulkopuolisten kantaa, jotka ovat voineet unohtua sähköpostin vastaanottajista.

Edellä mainittujen työkalujen lisäksi jokaisen osaprojektin sisällä ja kunkin järjestelmätoimittajan omassa toteutustyössä ja projektinhallinnassa voidaan käyttää myös muita työkaluja. Näiden osalta luvussa *3.3.14 Ohjelmistokehitys osaprojekteissa* kuvataan yrityksemme sisäisesti käyttämät työkalut osana asiakasyrityksen kokonaisprojektin toteutusta. Diplomityössä ei kuvata muiden osaprojektien käyttämiä työkaluja, vaan

keskitytään kokonaisprojektin hallinnassa yhteisesti käytettäviin työkaluihin, sekä yrityksemme sisäiseen projektinhallintaan ja kehitystyöhön.

3.3.8 Vaatimukset, liiketoimintaprosessit, määrittely

Vaatimukset ovat asiakasyrityksen tekemiä kuvauksia siitä, mitä toimintoja tietojärjestelmän tulee täyttää. Vaatimuksen laajuus vaihtelee sen mukaan, kohdistuuko se yksittäiseen toimintoon, vai kuvaako vaatimus isomman liiketoimintaprosessin. Liiketoimintaprosessit ovat yksittäisiä toimintoja laajempia kokonaisuuksia, jotka kuvaavat tietyn, useasta toiminnosta koostuvan prosessin suorittamisen. Liiketoimintaprosessi kohdistuu joko yhteen osajärjestelmään tai se voi vaatia useamman osajärjestelmän yhteistoimintaa. Osajärjestelmien välinen yhteistoiminta tarkoittaa järjestelmien integroimista toisiinsa erilaisin teknisin ratkaisuin. Lähtökohtana tässä on tietojärjestelmien välinen tiedonsiirto ja kommunikaatio.

Vaatimuksen ja liiketoimintaprosessin taustalla on määrittelytyö. Määrittely tehdään lähtökohtaisesti yhdessä asiakasyrityksen ja järjestelmätoimittajien kesken kunkin osaprojektin osalta. Määrittelyt, jotka koskettavat tietojärjestelmien välistä yhteistoimintaa tehdään yhdessä näiden tietojärjestelmien toimittajien ja asiakasyrityksen kanssa. Määrittelyllä pyritään löytämään paras mahdollinen ratkaisumalli, jonka pohjalta tietojärjestelmän toteutustyö voidaan aloittaa määritellyn toiminnon osalta. Määrittely, vaatimukset ja liiketoimintaprosessien kuvaukset esittävät sitä, mitä toteutettavalla kokonaisjärjestelmällä täytyy pystyä tekemään, jotta se täyttää asiakasyrityksen liiketoiminnan sille asettamat tarpeet.

Vaatimusten ja liiketoimintaprosessien dokumentointiin käytetään Confluencea. Nämä on jaettu wikissä loogiseen sivurakenteeseen, jossa kukin Confluencen pääsivu kuvaa tiettyä kokonaisuutta kuten tietojärjestelmän asiakastietoja, tuotetietoja tai myyntitilaus- ja ostotilausprosesseja. Näiden sivujen sisällöt tarkentavat kyseisiä käsitteitä niihin liittyvien vaatimusten, liiketoimintaprosessien ja käyttötapausten osalta. Confluenceen tuotetaan määrittelyiden tuloksena tekninen dokumentaatio, joka sisältää tarvittavan kuvauksen kyseisten vaatimusten ja prosessien tekniseen toteuttamiseen.

Confluenceen kuvatut vaatimukset ja liiketoimintaprosessit muodostavat tietojärjestelmän työlistan ketterien menetelmien mukaisesti. Vaatimusmäärittelyn yhteydessä käytetään käyttäjätarinoita kuvaamaan tietojärjestelmiltä odotettua toimintaprosessia tiettyjen toimintojen suorittamiseen käyttäjän näkökulmasta. Ketterille menetelmille on tyypillistä, että vaatimukset muuttuvat ja täydentyvät toteutustyön aikana. Tällaisessa tilanteessa muuttuneet vaatimukset kirjataan Confluenceen. Confluencessa on jokaisella sivulla versiohistoria, jonka kautta on nähtävissä kaikki vaatimuksiin ja kuvauksiin tehdyt muutokset.

3.3.9 Projektinhallinta, toteutustyön aikatauluttaminen

Jiraa käytetään kokonaisprojektin kehitystyön hallintaan ja toteutuksen seurantaan. Ohjelmisto on jokaisen projektiosapuolen käytössä. Jiraan on luotu jokaista osaprojektia kohden oma Jira-projekti. Näiden sisällä jokaisen osaprojektin tahot voivat tehdä suunnitelmia ja kehityksen seurantaa mikäli kokevat tämän tarpeelliseksi. Osaprojektikohtaisen projektinhallinnan käytännöistä sopivat järjestelmätoimittajat yhdessä asiakasyrityksen osaprojektista vastaavan henkilön kanssa. Käytännöt Jiran hyödyntämisen osalta vaihtelevat siis osaprojekteittain.

Kokonaishankkeen edistymisen osalta Jiraa käytetään kuitenkin yhteisesti tehtävienhallintaan ja projektisuunnitteluun. Jiraan on luotu osaprojektikohtaisten projektien lisäksi kokonaishankkeen eri toteutusvaiheita kuvaavat Jira-projektit. Näiden projektien sisällä seurataan toteutusvaiheen etenemistä yhteisesti kaikkien kyseiseen vaiheeseen osallistuvien järjestelmätoimittajien ja asiakasyrityksen kesken. Yrityksemme hyödyntää Jiraa tässä tarkoituksessa jokaista osapuolta koskettavan vaihetoteutuksen hallintaan. Yrityksemme ei käytä Jiraa osaprojektimme toteutuksen koordinoimiseen, vaan tässä hyödynnämme yrityksemme sisäisiä prosesseja, jotka on kuvattu luvussa *3.3.14 Ohjelmistokehitys osaprojekteissa*.

Jiraa käytetään toteutusvaiheiden seurannan osalta Scrumin periaatteiden mukaisesti: Projektin seuranta ja tehtävienhallinta on jaettu Jira-projektin sisällä sprintteihin. Sprintin kesto on noin yksi kalenterikuukausi. Confluenceen kuvatut liiketoimintavaatimukset ja käyttäjätarinat muodostavat tuotteen työlistan tikettien muodossa Jiraan kyseisen toteutusvaiheen osalta. Sprintin työlista suunnitellaan Jiraan sprintin suunnittelutapaamisessa tuotteen työlistasta. Ennen suunnittelutapaamista jokainen järjestelmätoimittaja on voinut käydä asiakasyrityksen kanssa yhteisesti läpi, toimittajan osajärjestelmän alustavaa suunnitelmaa seuraavan sprintin toteutettavista asioista. Alustavia suunnitelmia tehdään myös järjestelmätoimittajien välillä niistä toiminnoista, jotka vaativat tietojärjestelmien välistä yhteistoimintaa.

Alustavien suunnitelmien myötä sprintin suunnittelutapaamisessa jokaisella osapuolella on tietty näkemys sprintin sisällöstä heidän osaltaan. Tällöin suunnittelutapaamisen tehtävänä on yhteisesti hyväksyä, ja tehdä tarvittavat tarkennukset ja muutokset sprintin suunnitelmiin osajärjestelmittäin. Yhteisellä suunnittelulla pyritään tekemään sprintin työlistasta järkevä kokonaisuus, joka edistää mahdollisimman tehokkaasti toteutusvaiheen työlistaa. Sprintin toteutustyötä seurataan Jirasta paikallisten projektitapaamisten yhteydessä luvussa *3.3.11 Tapaamiset ja toteutuksen seuranta* kuvatun mukaisesti.

3.3.10 Laadunvarmistus, testitapaukset, hyväksymistestaus

Kokonaishankkeen onnistuneen lopputuloksen kannalta hankkeen laadunvarmistukseen on kiinnitetty erityistä huomiota. Laadunvarmistus on kiinteästi osana päivittäistä kehi-

tystyötä ja se näkyy Scrumin mukaisessa projektinhallinnassa. Jiraan määritellään jokaisen sprintin työlistalle testitapaustiketit, jotka kuvaavat Confluenceen kirjattuja käyttäjätarinoita. Testitapauksella on konkreettinen kuvaus järjestelmän käyttämisestä ja siitä, mitä toimenpiteitä järjestelmän käyttäjä tekee toteuttaakseen tavoittelemansa toiminnan. Projektin asiakasyritys vastaa testitapausten kirjaamisesta jokaisen sprintin aikana. Testitapaukset kirjataan niistä sprintin aikana kehitettävistä toiminnoista, joiden uskotaan valmistuvan sprintin aikana. Testitapausten avulla järjestelmätoimittajat voivat varmistaa toteuttamiensa tietojärjestelmien oikeanlaisen toiminnan jo kehitystyön aikana.

Jokaisen sprintin lopuksi pidetään sprintin jälkitarkastelu. Projektissa näitä tapaamisia kutsutaan hyväksymistestauksiksi. Hyväksymistestaus on yleensä kasvotusten pidettävä päivän mittainen tapaaminen, johon osallistuvat kaikki ne osapuolet, joilla on ollut sprintin aikana toteutustyötä, johon liittyy sprintin työlistalla olevia testitapauksia. Hyväksymistestauksessa käydään läpi sprintin aikana kehitetyt toiminnot, ja tarkastetaan vastaavatko nämä toiminnaltaan määriteltyjä vaatimuksia ja liiketoimintaprosessien kuvauksia. Hyväksymistestauksessa jokainen testattava toiminnallisuus käydään läpi testitapausten osoittamalla tavalla. Hyväksymistestauspäivän sisältö on tiedossa vähintään viikko ennen testausta. Testausta edeltävässä viikkopalaverissa varmistetaan kehitystyön tilanne ja lukitaan ne testitapaukset, jotka hyväksymistestauksessa tullaan suorittamaan.

Hyväksymistestauksen sisällössä on määritelty hyväksymiskriteerit jokaiselle testitapaukselle. Hyväksymiskriteerit on esitetty kolmella tasolla. Tasot ovat seuraavat:

1. Hyväksytty: Testitapaukselle määritetyt raja-arvot, joiden mukaisesti toiminnallisuus voidaan todeta hyväksytyksi. Käytännössä hyväksytty toiminta tarkoittaa sitä, että järjestelmä vastaa toiminnaltaan kaikkia testitapauksen kuvaamia vaatimuksia, eikä toiminta poikkea näistä.
2. Osittain toimiva: Testattava toiminnallisuus täyttää tietyt testitapauksen esittämät vaatimukset. Osa vaatimuksista jää täyttämättä tai järjestelmä toimii virheellisesti.
3. Hylätty: Toiminnallisuus ei täytä testitapauksen mukaisia vaatimuksia, toiminnossa havaitaan muita hyvin kriittisiä poikkeamia testitapauksen kuvaukseen, tai toiminnallisuutta ei voida testata hyväksymistestauksessa joko teknisen ongelman tai virheellisen toiminnan vuoksi.

Hyväksymiskriteereiden perusteella sprintin aikana toteutetut toiminnot joko hyväksytään, niitä joudutaan täydentämään tai ne hylätään. Hyväksymistestauksesta kootaan tulokset, jotka esittävät kriteerien täyttymisen. Tuloksien perusteella tehdään tarvittavat päätökset jatkotoimenpiteistä. Jos tulokset ovat erityisen huonoja, voidaan johtopäätöksenä toteuttaa välittömästi korjaava sprint, jonka aikana havaitut puutteet ja virheet korjataan ja toiminnot täydennetään vastaamaan testitapauksia. Toisena vaihtoehtona on aikatauluttaa tuloksissa havaitut puutteet ja virheet seuraavien sprinttien työlistoille.

Mikäli korjattavaa ja täydennettävää toiminnallisuutta on paljon, vaikuttaa tämä seuraavien sprinttien sisältöön lisätyötä aiheuttavana. Tällöin seuraavan sprintin suunnittelutapaamisessa täytyy huomioida edellisen sprintin tuloksista aiheutunut lisätyö, ja se, millä tavalla tämä vaikuttaa uuden sprintin työlistan toteuttamiseen ja sprintin aikatauluun.

Kokonaishankkeen osalta hyväksymistestaus suoritetaan erillisessä, tätä tarkoitusta varten olevassa testiympäristössä. Tietojärjestelmäkokonaisuudessa on kolme eri tietojärjestelmäympäristöä. Nämä ovat tuotanto-, testi- ja kehitysympäristöt. Hyväksymistestauksen tuloksena testiympäristössä hyväksytyt toiminnot voidaan päivittää hyväksymiskriteerien täyttämisen jälkeen järjestelmän tuotantoympäristöön, mikäli päivittämisestä on sovittu julkaisuaikataulun mukaisesti. Kehitysympäristöt on tarkoitettu jokaisen tietojärjestelmän uuden kehitystyön toteuttamiseen ja testaamiseen. Tutkimushetkellä kokonaishankkeen laajuudessa erillistä kehitysympäristöä ei ole vielä käytetty, vaan tietojärjestelmien välistä kehitykseen ja testaukseen on käytetty testiympäristöjä. Kehitysympäristöjen käyttäminen tulee tarpeelliseksi siinä vaiheessa, kun toteutusvaiheen laajempi käyttöönottotestaus aloitetaan. Käyttöönottotestaus suoritetaan toteutusvaiheen laajuudesta riippuen, viikosta reilun kuukauden mittaiseen hyväksymistestausjaksolla. Käyttöönottotestauksen yhteydessä käydään läpi toteutusvaiheen jokaisen sprintin testitapaukset läpi, sekä suoritetaan mahdollisia muita testauskokonaisuuksia kuten suorituskyky- ja tietoturvatestaukset. Tutkimushetkellä yrityksemme ei ole vielä osallistunut käyttöönottotestausjaksoon. Osajärjestelmämme osalta ensimmäinen käyttöönotto ajoittuu hankkeen aikataulun mukaisesti vuoden 2016 alkupuolelle, joten testausjakso suoritetaan osaltamme tätä ennen.

3.3.11 Tapaamiset ja toteutuksen seuranta

Projektissa järjestetään useita erilaisia tapaamisia, sekä kasvotusten että videon välityksellä. Yleensä osallistujat voivat valita kummalla tavalla osallistuvat tapaamiseen, mutta jokaisella toimittajalla on vastuu miettiä onko tapaamiseen paikan päälle tuleminen järkevämpää, vai pystyykö videon välityksellä osallistumaan tarpeeksi hyvin. Käytännöt tapaamiseen osallistumisesta vaihtelevat tapauskohtaisesti palaverin luonteen ja sisällön mukaan. Tapaamisen laajuus vaihtelee tapahtuman sisällön mukaan: tietyt tapaamiset voivat olla kahden osapuolen välisiä, kun toisissa voi olla mukana kaikki järjestelmätoimittajat ja asiakasyrityksen edustajat.

Seuraavassa kuvataan tapaamiskäytännöt tutkimuskohteen operatiivisen tason toteutusvaiheessa, jossa yrityksemme on mukana tutkimushetkellä.

Sprintin suunnittelutapaaminen on jokaisen sprintin alussa pidettävä tapaaminen, jossa suunnitellaan sprintin työlista ja sovitaan sprintin alustavasta aikataulusta. Sprintin suunnittelutapaaminen järjestetään yleensä videon välityksellä, mutta jos sprintin sisältö on erityisen laaja, voidaan osapuolien kesken sopia suunnittelusta kasvotusten. Jos edel-

tävä sprint on ollut työlistaltaan pieni, voidaan seuraavan sprintin suunnittelu järjestää edeltävän sprintin hyväksymistestauksen yhteydessä.

Sprintin statuspalaveri on viikoittain järjestettävä videopalaveri, johon osallistuu vähintään jokaisen järjestelmätoimittajan projektipäällikkö. Palaverissa käydään läpi sprintin työlistalla olevien tehtävien tilanne kunkin järjestelmätoimittajan osalta. Tilanne käydään läpi Jirassa olevan Kanban-näkymän kautta. Kanban-näkymässä nähdään kaikki Jirassa olevat tehtävät visuaalisesti niiden toteutusvaiheiden mukaisesti. Tehtävien tilanne käydään läpi näkymässä järjestyksessä ylhäältä alas. Jokaisen tehtävän kohdalla keskustellaan tilanne tehtävään liittyviltä osapuolilta, ja varmistetaan, että tehtävän tilanne on ajan tasalla ja toteuttaminen voi jatkua. Statuspalaverin tarkoituksena on tuoda ilmi mahdolliset ongelmat tai haasteet, joita tehtävän suorittamiseen liittyy.

Kanban-näkymän ohella, toinen käytössä oleva tapa statuspalaverin läpikäyntiin, on käydä jokaisen järjestelmätoimittajan työlistan tilanne läpi toimittaja kerrallaan. Tätä käytäntöä on hyödynnetty etenkin silloin, jos sprintin työlista on niin suuri, että sen läpikäyminen Kanbanin kautta olisi turhan hidasta. Sprintin statuspalaverin kesto on maksimissaan kaksi tuntia. Tilanteen läpikäyntitapa sovitaan jokaisen palaverin alussa, ja se voi olla myös mainittujen yhdistelmä: toinen järjestelmätoimittaja voi käydä oman tilanteensa läpi esimerkiksi oman listauksensa mukaan, kun toisen osalta voidaan rajata Jiran Kanbaniin näkyville vain tietyn osajärjestelmän tehtävät.

Päivästatus on *"Daily Scrumin"* mukainen statuspalaveri, jossa käydään läpi järjestelmätoimittajittain päivän toteutustyön tilanne. Jokainen järjestelmätoimittaja kertoo tilanteen heillä kyseisen päivän aikana toteutettavista tehtävistä. Tarkoituksena on lisäksi tuoda ilmi mahdolliset ongelmat ja haasteet, jotka vaikuttavat tehtävien toteuttamiseen päivän aikana. Päivästatus pidetään aamulla Flowdockin välityksellä tähän tarkoitetussa huoneessa.

Päivästatukset eivät ole projektissa aktiivisessa käytössä, mutta niitä on hyödynnetty, jos niiden käyttö on sprintin statuspalaverissa koettu tarpeelliseksi. Päivästatuksia on pidetty etenkin silloin, jos sprintin toteuttamiseen liittyy erityisen paljon riippuvuuksia järjestelmätoimittajien väliseen työskentelyyn, tai jos sprintin aikataulu vaatii erityistä tarkkuutta toteutuksen sujuvaan etenemiseen. Riippuvuuksilla tarkoitetaan tässä yhteydessä sitä, että jotta osajärjestelmän työtä voidaan kyseisenä tai tulevina päivinä edistää, vaatii se jonkin toisen järjestelmätoimittajan työpanosta ennen tätä.

Määrittely- ja suunnittelutapaamiset ovat kahden tai useamman osapuolen välisiä tapaamisia, joissa käydään läpi tietojärjestelmiin toteutettavia toimintoja. Tapaamisten sisältö vaihtelee yleisestä vaatimusten ja asiakastarpeiden läpikäymisestä sekä suunnittelusta, todella tarkkaan tekniseen toteutuskeskusteluun. Tapaamisten käytäntöihin ei ole yleisiä ohjeita, vaan ne voivat poiketa toisistaan suuresti. Tapaamisia järjestetään aina tarvittaessa, ja niistä voidaan sopia osapuolien kesken hyvin joustavasti. Projektin

aikana on havaittu, etenkin uusien vaatimusten ja asiakastarpeiden suunnittelemisen yhteydessä, että kasvotusten tapaaminen johtaa yleensä parempaan lopputulokseen, sillä epäselvät ja hankalat asiat on helpompi käsitellä kasvotusten.

3.3.12 Dokumentointi

Hankkeen dokumentaation keskitettyyn hallintaan käytetään Confluencea. Dokumentaatioon on kirjattu osajärjestelmiä koskettavaa informaatiota seuraavista näkökulmista.

1. Vaatimukset kuvaavat asiakasyrityksen määrittelemät liiketoimintavaatimukset osajärjestelmille.
2. Käyttötapaukset kuvaavat järjestelmille tehdyt toimintakuvaukset siitä, kuinka yksittäisiä toimenpiteitä suoritetaan loppukäyttäjän näkökulmasta järjestelmän sisällä.
3. Tekniset kuvaukset järjestelmien yksittäisten toimintojen toiminnasta, ja siitä, kuinka kyseiset toiminnot on toteutettu.
4. Tekniset kuvaukset järjestelmien välisten integraatioiden prosessikuvauksista sekä tietovirroista ja -sisällöistä.

Näiden lisäksi Confluencessa kuvataan järjestelmiin liittyvä julkaisuaikataulu ja tulevat päivitykset, joita tehdään tuotantojärjestelmiin. Yksittäisten julkaisujen yhteydessä dokumentoidaan mitä uutta toiminnallisuutta, vanhojen toimintojen muutoksia ja virheiden korjauksia päivitys sisältää. Sivustolle tallennetaan myös tapaamisten muistiinpanot, jolloin ne ovat keskitetysti kaikkien osapuolien saatavilla. Muistiinpanojen yhteyteen voidaan kirjata tehtävälistoja sovituista toimenpiteistä, mitä tiettyjen henkilöiden tai osapuolien täytyy tapaamisen jälkeen sovitusti hoitaa.

Tarvittavasta dokumentoinnista ja muistiinpanojen kirjaamisesta sovitaan tapaamisten yhteydessä. Dokumentaation päivittämisestä ja kirjaamisesta voidaan sopia myös sprintin tehtävien yhteydessä Jiran kautta.

3.3.13 Tukitoiminnot

Projektin- ja tehtävienhallinnan lisäksi Jiraa käytetään tuotannossa olevien tietojärjestelmien tukipyyntöjen ja palautteen, sekä järjestelmävirheiden kirjaamiseen sekä raportointiin. Näitä molempia varten on Jirassa omat projektinsa, joihin tiketit luodaan. Tukipyynnöt kirjataan asiakasyrityksen toimesta ja järjestelmätoimittaja vastaa niihin yhdessä asiakasyrityksen kanssa sopimallaan tavalla. Palautteet huomioidaan asiakasyrityksen ja järjestelmätoimittajan kesken kyseisen tietojärjestelmän suunnittelussa, ja niistä voidaan kirjata tiketit osajärjestelmän työlistaan, josta ne voidaan ottaa toteutukseen sovitun aikataulun mukaisesti. Virhetiketit on jaettu niiden vakavuuden perusteella kolmeen osaan:

1. Korkean tason vakavuus. Tämä taso sisältää välittömästi korjausta vaativat virheet, joiden korjaaminen priorisoidaan virheen kohteena olevien järjestelmien kaiken muun kehitystyön edelle. Tällaisten virheiden korjaukset on päivitettävä tuotantojärjestelmään välittömästi korjauksen valmistuttua.
2. Keskitason vakavuus. Tämä taso sisältää seuraavaan päivitysversioon korjattavat virheet, jotka viedään tuotantoon järjestelmän julkaisuaikataulun mukaisesti seuraavan päivityksen yhteydessä. Näiden virheiden korjaus aikataulutetaan osajärjestelmien kehitystyöhön tarvittavan prioriteetin mukaisesti, jotta korjaus saadaan toteutettua ennen seuraavaa tuotantopäivitystä.
3. Matalantason vakavuus. Alimman tason virheet priorisoidaan ja päivitetään tuotantojärjestelmään ennen seuraavaa asiakasyrityksen liiketoimintaan kohdistuvaa kausiluontoista sesonkia. Tämän hankkeen osalta asiakasyrityksen sesongit ajoittuvat vuosittain kevääseen ja loppusyksyyn autojen renkaanvaihtokausien ja säätilanteen mukaisesti.

3.3.14 Ohjelmistokehitys osaprojektissa

Osajärjestelmäkohtaiset toimintatavat ja ohjelmistokehitysmenetelmät poikkeavat toisistaan kunkin järjestelmätoimittajan toimintatapojen mukaan. Tässä yhteydessä tapaustutkimuksen näkökulmasta ei ole tarpeen käydä läpi muiden järjestelmätoimittajien prosesseja, vaan kuvataan yleisellä tasolla yrityksemme ohjelmistokehityksen menetelmät.

Yrityksemme tuotekehitys ja asiakasyritykselle toteutettavan osajärjestelmän kehitys on tehty toisiaan tukeviksi prosesseiksi. Kehitämme asiakasyrityksen osajärjestelmää yhdessä yrityksemme oman tuotteen kanssa, sillä näiden välillä on toimintojen osalta vain pieniä asiakasyritykselle tehtäviä kustomoituja ominaisuuksia. Yrityksemme sisäiset toimintatavat ja kehitysprosessit tukevat asiakasyrityksen projektin toteuttamista. Käytämme sisäisesti ketteriä ohjelmistokehitysmenetelmiä samalla tavalla kuin asiakasyrityksen hankkeessa. Tämä on luontevaa, koska ketterät menetelmät otettiin hankkeessa käyttöön nimenomaan yrityksemme aloitteesta.

Yrityksessämme on käytössä Scrumin mukainen sprint-ajattelu. Scrumia ei noudateta täydellisen tarkasti, vaan sitä käytetään samalla tavalla kuin tässä luvussa on tapaustutkimuksen osalta todettu hankkeessa käytettävän. Scrumin mukaisen kehitystyön hallinta ja seuranta tehdään Kanban-näkymien avulla. Tähän tarkoitukseen käytetään Trellon (2015) kehittämää työkalua, joka visualisoi Kanbanin mukaisen työnkulun seurannan. Yrityksemme tekee hankkeen sprint-suunnitelmat ensin sisäisesti Trelloon. Suunnitelmien yhteydessä pyritään määrittelemään toteutettavat toiminnot tarpeeksi tarkalle tasolle, jotta niiden toteutus voidaan aloittaa. Trellossa tehty suunnitelma siirretään projektipäällikön toimesta asiakasyrityksen Jiraan tarvittavilta osin; käytännössä vähintään niiden tehtävien osalta, jotka koskettavat hankkeen muita osajärjestelmiä. Yrityksemme tehtävien edistymistä seurataan sisäisesti Trellosta maanantaisin pidettävien viikkostatuksien yhteydessä, sekä päivittäin kehitystyön edetessä ja Scrumin mukaisten *“Daily*

Scrum” -palaverien yhteydessä. Jirassa olevilla tehtävillä kommunikoidaan niihin liittyvä toteutustyö muiden järjestelmätoimittajien suuntaan.

Yrityksemme käyttää myös sisäisesti kommunikointiin Flowdockia. Samojen työkalujen käyttäminen yksinkertaistaa yrityksemme osalta hankkeen toteuttamista, sillä Flowdockin kautta saamme yhteyden sekä yrityksemme omiin tiimeihin että hankkeen muihin osapuoliin. Yrityksemme on jaettu ohjelmistokehittäjien osalta kahteen tiimiin, jotka molemmat hyödyntävät ketteriä menetelmiä samalla tavalla. Näistä tiimeistä toinen toteuttaa asiakasyritykselle tehtävää hanketta, ja toinen keskittyy yrityksemme sisäisten palveluiden kehittämiseen.

Yrityksemme sisäiset toimintatavat tukevat hankkeen toteutusta jokaisen osa-alueen kannalta. Tiimimme on organisoitu ja jaettu sillä tavalla, että hankkeen edellyttämiin määrittely- ja suunnittelutilaisuuksiin, sekä muihin tapaamisiin osallistuu tilanteen mukaisesti tarvittavat henkilöt. Asiakasyrityksen kuvaamat liiketoimintavaatimukset ja käyttäjätarinat käydään läpi yrityksemme toteutustyön yhteydessä. Hankkeen toteutusvaiheiden mukaiset hyväksymistestaukset, ja niiden sisältämät tehtävät, käydään sisäisesti läpi ja osajärjestelmän oikea toiminta testataan ennen projektiosapuolien välistä hyväksymistestaustapaamista. Scrumin mukaisesti yrityksemme toteutustyön dokumentointiin ei kiinnitetä liian suurta tarkkuutta. Dokumentoinnin osalta pyritään varmistamaan, että toteutettava osajärjestelmä täyttää Confluenceen kirjatut vaatimukset ja määritelmät, mutta näiden ohelle ei sisäisesti tarvitse toteuttaa toisteista dokumentaatiota.

4. KETTERIEN MENETELMIEN SOVELTUVUUS USEAN JÄRJESTELMÄTOIMITTAJAN IT-HANKKEESEEN

Tässä luvussa esitetään diplomityön tulokset analysoimalla suoritettua tutkimusta. Tulosten yhteydessä esitetään arvioita ketterien ohjelmistokehitysmenetelmien käytöstä, tuodaan esille niiden hyviä puolia sekä havaittuja ongelmia ja haasteita. Luvussa 4.2 arvioidaan saavutettujen tulosten luotettavuutta ja käyttöarvoa, sekä pohditaan mahdollisia jatkotutkimuskohteita.

4.1 Case-tutkimuksen tulokset

Seuraavassa esitettävät tapaustutkimuksen tulokset on saavutettu tutkimuskohteen havainnoinnin yhteydessä. Hirsjärvi et al. (2007) mukaan on tavallista, että kvalitatiivisessa tutkimuksessa analyysia ei tehdä vain yhdessä tutkimusprosessin vaiheessa, vaan pitkin matkaa tutkimuksen suorittamisen aikana. Aineistoa siis kerätään ja analysoidaan samanaikaisesti. Tämä on tyypillistä etenkin osallistaville kenttä- ja tapaustutkimuksille, joissa havainnointi tehdään pääosin tapahtumapaikalla. (Hirsjärvi et al. 2007)

4.1.1 Tavoitteet, asiakastarpeen ymmärtäminen

Tietojärjestelmähankkeen onnistumisen kannalta on tärkeää, että jokainen osapuoli tietää ja tunnistaa ne lähtökohdat, joiden vuoksi hanke on käynnistetty. Nämä lähtökohdat määrittelevät hankkeelle tavoitteet, joiden ymmärtäminen mahdollistaa onnistuneen lopputuloksen saavuttamisen. Arvot luovat pohjan osapuolien väliselle toiminnalle, ja asetettujen tavoitteiden ymmärtäminen on tärkeää hankkeen toteuttamiselle.

Haastavaksi tavoitteiden ja arvojen sisäistämisen tekee se, ymmärtävätkö kaikki osapuolet ne asiakkaan tarkoittamalla tavalla. Tämä on syytä varmistaa asiakkaan toimesta aina hankkeen alkuvaiheessa, sillä jos jokaisella järjestelmätoimittajalla on samanlainen näkemys toivotusta lopputuloksesta, antaa se varmemman pohjan sille, että hankkeessa tehdään oikeita ratkaisuja asetettujen tavoitteiden kannalta. Suuressa ja pitkäkestoisessa projektissa olisi hyvä järjestää osapuolien välille myös tavanomaisista projektitapaamisista poikkeavia, vapaamuotoisia tapahtumia. Tällaisten tapaamisten nähtiin parantavan tutkimuksen aikana ymmärrystä osapuolien välillä. Vapaamuotoinen keskustelu osana tietojärjestelmähanketta tarjosi osapuolille mahdollisuuden ymmärtää toistensa toimintatapoja sekä yksittäisiin osajärjestelmiin liittyviä tavoitteita ja vaatimuksia. Normaalei-

den projektitapaamisten yhteydessä järjestelmätoimittajille ei välttämättä muodostunut kovin kattavaa kuvaa toistensa toteuttamista osajärjestelmistä, vaan näkökulma tietojärjestelmien väliseen yhteistyöhön jäi melko pintapuoleiseksi. Vapaampi keskustelu edesauttoi näkemään hankkeen selkeämmin kokonaisuuden kannalta, mikä mahdollistaa toteutuksen yhteydessä parempien toimintatapojen ja ratkaisuiden löytämisen.

Liiketoimintavaatimusten kirjaaminen tarpeeksi tarkalle tasolle on edellytys tietojärjestelmien tehokkaalle toteuttamiselle. Vaatimuksista järjestelmätoimittajat pystyvät ymmärtämään asiakkaan tarpeet. Usean järjestelmätoimittajan yhteistyössä on liiketoimintavaatimusten yhteydessä käytävä läpi erityisen tarkasti tietojärjestelmien väliset riippuvuudet. Osapuolien on tunnistettava eri järjestelmien vastuut, ja asiakasymmärryksen kautta määriteltävä, mitä kunkin tietojärjestelmän toimintoihin kuuluu. Vastuualueiden määrittäminen tietojärjestelmien välille on tärkeää, sillä ilman rajausta tietojärjestelmät voivat olla riippuvaisia toisistaan tarpeettoman paljon, sekä asettaa toisilleen lisävaatimuksia, joille ei välttämättä ole todellista liiketoimintatarvetta.

Liiketoimintavaatimuksista muodostuu toteutettavan tuotteen työlista. Hankkeen luonteesta riippuen, työlista voi olla kokonaisuudessaan tiedossa, kun hanke aloitetaan, tai työlista voi muodostua ja täydentyä vasta hankkeen edetessä. Usein todellisuudessa tietojärjestelmähankkeet ovat todennäköisesti näiden yhdistelmä, jossa tietyt esivaatimukset ovat tiedossa, mutta tarkemmat kuvaukset uusien ominaisuuksien toiminnasta muodostuvat vasta määrittelytapaamisten yhteydessä. Määrittelytapaamisiin on kiinnitettävä erityistä huomiota, jos aiotaan toteuttaa uusia toimintoja tai uudistaa vanhoja. Usean osapuolen välisessä hankkeessa tämä on hyvin kriittistä, sillä tehokkaan toteuttamisen kannalta on pyrittävä muodostamaan osapuolille yhtenäinen näkemys tarpeista. Ristiriitaisuudet ja epäselvyydet voidaan pahimmassa tapauksessa huomata vasta siinä vaiheessa, kun tietojärjestelmät ovat tuotantokäytössä. Tällaisesta tilanteesta aiheutuvat korjausvaatimukset ja muutostyöt aiheuttavat merkittävää haittaa hankkeen toteutuksen normaalille etenemiselle, sillä ne vaativat aikaa ja varaavat toteutukseen kohdistettuja resursseja ennalta-arvaamattomien tehtävien tekemiseen.

Tapaamiskäytännöistä on sovittava osapuolten kesken selkeästi. Ketterät menetelmät pitävät oleellisena periaatteena sitä, että keskitytään olennaiseen. Liiketoimintavaatimusten kartoittaminen määrittelytapaamisten avulla on oltava organisoitua. Määrittelyyn on syytä osallistua vain ne osapuolet, joihin määrittelyt vaikuttavat. Tämä edesauttaa osapuolten keskittymistä oleelliseen, kun tapaamisiin ei sidota ylimääräisiä henkilöitä turhaan. Tarpeiden ja vaatimusten kartoittamisen kannalta on hyvä sopia myös käytössä olevien työkalujen merkityksestä ja käyttöperiaatteista. Liiketoimintavaatimusten kirjallinen dokumentointi on tärkeää, jotta tietojärjestelmän toteutusvaiheessa tarvittava informaatio on saatavilla. Dokumentointikäytännöstä on sovittava yhteisesti, ja kaikki tarvittava materiaali on hyvä tallentaa niin, että jokainen osapuoli tietää mistä informaatio on löydettävissä, ja osapuolet voivat luottaa tiedon oikeellisuuteen.

Dokumentaation hallintaan liittyy omat haasteensa, kun informaatio on usean osapuolen saatavilla ja mahdollisesti myös muokattavissa. Osapuolien täytyy sopia miten varmistetaan informaation oikeellisuus ja ajantasaisuus. Toimintatapojen yhteydessä on lisäksi varmistettava, ettei päällekkäistä tai ristiriitaista informaatiota synny. Osapuolien hieman toisistaan poikkeavat dokumentointikäytännöt voivat aiheuttaa saman tiedon tallentamisen useaan paikkaan, jolloin lopputuloksena voi syntyä tiedon ristiriitaisuutta. Huonossa tilanteessa ristiriitaisuudet voivat tulla ilmi vasta sprintin jälkitarkastelun yhteydessä, kun toiminnallisuuksien toteutusta hyväksytään. Dokumentointikäytäntöjen yhteydessä on sovittava vastuut, toimenpiteet ja käytävä läpi informaation merkityksellisyys, jotta yhteiset käytännöt tulevat tehokkaaseen käyttöön jokaisen osapuolen toimesta, ja niitä myös noudatetaan sovitusti.

4.1.2 Projektinhallinta, toteutuksen aikatauluttaminen ja seuranta

Usean osapuolen välisessä kehitystyössä, jossa tietojärjestelmillä on riippuvuuksia toisiaan kohtaan, on erittäin tärkeää, että osapuolien välinen kehitystyö etenee toistensa rinnalla. Tietojärjestelmien toteutuksen aikatauluttamisen kannalta jokaisella osapuolella täytyy olla selkeä tilannekuva toteutuksen etenemisestä ja vaiheistuksesta. Sprinttien suunnitteluun ja seurantaan on kiinnitettävä erityistä tarkkuuta, ja jokaisen järjestelmätoimittajan täytyy pystyä tuomaan ilmi toimenpiteet ja tehtävät, joita heidän kehitystyönsä vaatii muilta osapuolilta. Näillä toimintatavoilla voidaan hallita tietojärjestelmien välisiä riippuvuuksia, ja niiden kehityksen tehokasta yhdistämistä.

Scrumin mukaiset sprintit vaiheistavat hankkeen toteuttamista. Tämä helpottaa toteutusvaiheiden seurantaa, tehtävien priorisointia ja sprinttien sisällön suunnittelua. Sprintin suunnittelussa jokaisen järjestelmätoimittajan täytyy arvioida sprintin työlistalta oman osuutensa vaatima työmäärä. Arviot käydään yhteisesti osapuolien välillä läpi, jolloin havaitaan pystytäänkö sprintin työlistan mukaiset tehtävät toteuttamaan sprintin aikana. Sprintin suunnittelukäytännöt mahdollistavat sprintin työlistan rakentamisen tuotteen työlistalla olevista tehtävistä, yhteistyössä osapuolien kesken. Tämä tuo näkyväksi tietojärjestelmien toteutustyötä tekeville osapuolille seuraavan sprintin sisällön, ja määrittelee minkä tehtävien suorittamiseen sprintin aikana sitoudutaan.

Scrumin mukaisesti sprintin sisältöön ei saa sen aikana tulla uusia tehtäviä. Uudet vaatimukset voidaan kirjata muistiin, mutta ne tuodaan toteutukseen aina vasta seuraavia sprinttejä suunniteltaessa. Tähän täytyy kiinnittää huomiota, sillä ylimääräinen tai muuttunut tehtävä aiheuttaa tietojärjestelmän kehityksessä sen, että toteutus täytyy suunnitella uudelleen, mikä tarkoittaa sprinttiin vaaditun työmäärän kasvua. Yksittäisen sprintin sisällä tapahtuva työmäärän kasvu, vaikuttaa hyvin todennäköisesti myös tulevien sprinttien työmääriin ja aikatauluihin. Tutkimuksen yhteydessä tämä ongelma havaittiin

todella yleiseksi, sillä lähes jokaisen sprintin yhteydessä työlista päivittyi tai siihen tuli uusia tehtäviä sprintin aikana.

Mikäli jokin työlistan tehtävä ei ole tiedossa sillä tarkkuudella, että se voitaisiin tietojärjestelmän toiminnallisuuden osalta toteuttaa, on sprintin työlistalle tarpeen kirjata tehtävä, jolla kyseinen toiminnallisuus määritellään sprintin aikana tarkemmalle tasolle. Tällöin kyseinen toiminnallisuus voidaan ottaa seuraavassa sprintissä toteutukseen, kun vaatimus on määritetty tarpeeksi tarkaksi, eikä siihen liity epäselviä asioita. On todella tärkeää, että vaatimusten ja tarpeiden määrittelystä ja suunnittelusta on kirjattu tehtävät erikseen. Tällöin toiminnallisuuden suunnittelu ja toteuttaminen voidaan pitää omina tehtävinään, ja nämä voidaan aikatauluttaa sprinttien työlistoille tehokkaammin.

Usean osapuolen välisen hankkeen toteutuksen seuranta vaatii sovittujen käytäntöjen ja työkalujen aktiivista hyödyntämistä, jokaisen osapuolen toimesta. Käytössä olevien työkalujen tarjoama tuki kannattaa hyödyntää. Kanban-näkymän ideana on esittää jatkuvasti ajantasainen kuva tehtävien tilanteesta. Tehokkaan ja aktiivisen seurannan kannalta on olennaista, että Kanban-näkymä on osapuolien saatavilla. Työkalujen käyttötavat, käytön aktiivisuus ja osaaminen vaihtelevat osapuolien välillä, jolloin todellisen kokonaiskuvan selkeä hahmottaminen voi olla haastavaa. Kokonaistilanteen hahmottaminen usean järjestelmätoimittajan hankkeessa asettaa jokaiselle osapuolelle vastuun noudattaa sovittuja käytäntöjä, jotta hankkeen edistymisen osalta saatavilla olevaan informaatioon toteutuksen tilanteesta voidaan luottaa.

4.1.3 Laadunvarmistusprosessi, asiakastyytyväisyyden saavuttaminen

Scrumin mukaisesti sprintin jälkitarkastelussa on tarkoitus esitellä ainoastaan täysin valmiita toiminnallisuuksia. Tietyissä tapauksissa asiakas voi haluta nähdä toteutuksessa olevien toiminnallisuuksien tilanteen, mutta keskeneräisten toimintojen esittämiseen jälkitarkastelu ei ole oikea tilanne. Kuten edellisissä luvuissa kuvattuihin tapaamisten, vaatimusmäärittelyjen ja projektinhallinnan käytäntöihin, niin myös laadunvarmistuksen osalta on sovittava selkeät käytännöt osapuolien kesken. Keskeneräisen toiminnallisuuden läpikäyntiin on sprintin työlistalla varattava tarvittava aika, mikäli läpikäynnin tiedetään olevan tarpeen sprintin suunnittelun yhteydessä. Toinen vaihtoehto on varata jokaisen toteutettavan tehtävän työmääräarvioon pieni osuus toiminnallisuuden läpikäymiseen ja mahdollisten tarkennuksien tekemiseen. Keskittymällä jälkitarkastelussa vain valmiiden toimintojen testaukseen, tehostetaan tällä tapaamisen kulkua, ja vältytään aikaa vievältä, jo etukäteen tiedossa olevien puutteiden ja huomioiden kirjaamiselta. Tutkimuksen yhteydessä havaittiin, että varsinkin ketterien menetelmien käyttöönottamisen alkuvaiheessa, keskeneräistä toiminnallisuutta esitettiin jälkitarkastelussa todella paljon. Prosessin kehittyessä ja tutkimuksen edetessä tähän on reagoitu paremmin, ja Scrumin mukaista periaatetta on alettu noudattamaan paremmin.

Vaikka ketterät menetelmät mahdollistavat muutoksiin reagoimisen, on muutoksen kuitenkin oltava hallittua. Havaittu muutostarve ei saa vaikuttaa välittömästi tietojärjestelmän sen hetkiseen toteutustyöhön. Scrumissa on erittäin tärkeää, että sprintin työlista on kiinnitetty. Työlista hyväksytään sprintin suunnittelutapaamisessa, ja siihen ei sprintin aloituksen jälkeen tehdä muutoksia tai lisäyksiä. Mikäli työlistalla olevan toiminnallisuuden toteuttamiseen kohdistuu sprintin aikana muutosvaatimus, on siinä tapauksessa tarpeen kiinnittää huomiota sprintin suunnitteluun ja siihen, kuinka suunnitteluvaiheessa työlistalle kiinnitetty tehtävä ei ole ollut määrittelyltään vaadittavalla tasolla, jotta toteutustyö sen osalta voidaan aloittaa ja saattaa valmiiksi.

Millä tavalla sprintin suunnittelussa voidaan siis onnistua? Tärkein asia sprintin suunnittelussa on tiedostaa mitä ollaan tekemässä ja mitä tehtäviä ylipäänsä voidaan tehdä. Sprintin työlistalle voidaan ottaa tietojärjestelmän toteutusmielessä vain sellaisia tehtäviä, jotka ovat määrittelyltään sillä tasolla, ettei niiden toteuttamiseen liity tarkennusta vaativia asioita. Ohjelmistokehityksessä ei voida varmuudella arvioida sellaisen tehtävän toteuttamiseen kuluvaa aikaa, mitä ei ole tarkasti määritelty tai johon liittyy epävarmuustekijöitä. Usean järjestelmätoimittajan hankkeessa on erityisen tärkeää kiinnittää huomiota todenmukaisten työmääräarvioiden tekemiseen, jotta sprintin toteuttamisessa voidaan onnistua ja tietojärjestelmien välisiä riippuvuuksia pystytään hallitsemaan. Mikäli työmääräarvio arvioidaan liian pieneksi, täytyy muista sprintin tehtävistä suoriutua nopeammin tai jättää jotain tehtäviä toteuttamatta. Virheellinen työmääräarvio voi usean toimittajan tapauksessa, vaikuttaa negatiivisesti usean tietojärjestelmän kehityksen edistymiseen. Jos jonkin osapuolen toimesta työmäärä on arvioita virheellisesti, tämä vaikuttaa riippuvuuden myötä muiden toimittajien aikatauluihin.

Jos toiminnallisuudesta on kirjattu käyttäjätarina tai testitapaus, niin jokainen näiden sisältämä asia täytyy olla tarkasti selvillä, ennen kuin kyseinen tehtävä voidaan ottaa sprintin työlistalle toteutettavaksi. Testitapauksien sisällön on hyvä olla laajuudeltaan mieluummin liian pieniä kuin liian suuria. Tarpeeksi pienet ja tarkat testitapaukset varmistavat sen, että niiden toteuttaminen on tehokasta. Määrittelemällä testitapaukset tarkoiksi ja laajuudeltaan mahdollisimman pieniksi osiksi, saadaan laajatkin toiminnallisuudet toteutettua hallitusti vaiheistettuna. Selkeisiin osiin kirjatut testitapaukset tukevat erittäin hyvin Scrumin mukaista inkrementaalista toimintojen kehittämistä. Lisäksi todenmukaisen työmääräarvion tekeminen on järjestelmätoimittajien osalta helpompaa tarkasti määritetyille testitapauksille. Epävarmuutta ja tarkennuksia vaativat testitapaukset on syytä siirtää seuraavan sprintin toteutukseen, ja sopia, että tarvittavat tarkennukset tehdään toteutusta edeltävän sprintin aikana.

Onnistuneesti suunniteltu, arvioitu ja toteutettu sprint takaa asiakkaan tyytyväisyyden hankkeen etenemiseen, sekä edesauttaa varmistamaan tietojärjestelmähankkeen laadukkaan lopputuloksen. Onnistunut sprint toteutetaan tehdyn suunnitelman mukaisesti, eikä mikään työlistalle valittu tehtävä jää suorittamatta epäselvyyksien tai muuttuneiden vaa-

timusten vuoksi. Onnistuneen sprintin lähtökohtana on yhteistyö, kommunikointi ja selkeät toimintatavat suunnitelmia tehtäessä.

4.1.4 Ketterien menetelmien hyödyntäminen

Ketterien menetelmien käyttämisellä usean järjestelmätoimittajan tietojärjestelmähankkeessa voi olla erittäin merkittävä vaikutus hankkeen lopputulokseen. Käytettävästä menetelmästä tai usean menetelmän yhdistelmästä sovittaessa on tärkeää käydä läpi toimintatavat ja valitut prosessit hankkeen toteutukseen projektinhallinnallisesta näkökulmasta. Scrumin mukaiset sprintit koordinoivat ja vaiheistavat osapuolien välistä työskentelyä. Tämä edesauttaa tehostamaan kunkin osajärjestelmän toteuttamista.

Suuren hankkeen vaiheistaminen on kriittistä, sillä ohjelmistokehityksessä on tärkeää pystyä keskittymään yhteen asiaan kerrallaan. Scrumin vaiheistusmalli tarjoaa sprinttien muodossa monipuolisia toimintatapoja myös usean järjestelmätoimittajan väliseen yhteistyöhön. Sprintit eivät vaiheista ainoastaan hankkeen toteutustyötä, vaan ne määrittelevät myös käytännöt uusien toiminnallisuuksien ja liiketoimintatarpeiden selvittämiseen, suunnitteluun ja toteutustyön aikatauluttamiseen.

Tutkimuksen yhteydessä havaittiin, että asiakkaan voi olla vaikeaa hahmottaa ketterien menetelmien periaatteet, mikäli niiden hyödyntämisestä ei ole aiempaa kokemusta. Scrumin mukainen inkrementaalinen kehitystapa voi aiheuttaa asiakkaalle tunteen, että samoja asioita käsitellään useita kertoja, vaikka toimintatapa on hyvin olennainen osa ketterää ohjelmistokehitystä. Ketterien menetelmien periaatteiden ymmärtäminen on siten tärkeää myös asiakkaan osalta.

Lisäksi vaiheistuksen myötä keskittyminen olennaiseen ja turhan työn välttäminen ovat olennaisia piirteitä ketterille toimintatavoille. Mikäli asiakas on tottunut perinteisempiin projektinhallintamalleihin, voi tällainen toimintatapa vaikuttaa siltä, että liiketoimintavaatimusten määrittely ja läpikäynti siirtyy asiakkaan näkökulmasta liian myöhäiseen vaiheeseen hanketta. Ketterien menetelmien osalta on ymmärrettävä, että sprinttien vaiheistaminen mahdollistaa asioiden tehokkaan priorisoinnin. Priorisointi on syytä olla näkyvillä, jotta asiakas hahmottaa missä vaiheissa liiketoimintavaatimuksia käsitellään. Kaikkien vaatimuksien läpikäyminen hankkeen alkuvaiheessa ei ole järkevää, sillä hanke voi olla kestoltaan hyvin pitkä, jolloin tietyt vaatimukset voivat hankkeen toteutusaikana jo vanhentua ajan myötä. Tehokkaana keinona on pyrkiä priorisoimaan hankkeen alkuvaiheen sprintteihin sellaisten vaatimusten ja asiakastarpeiden toteutus, joiden voidaan varmimmin sanoa pysyvän muuttumattomina. Tällöin tehtävät, jotka ovat epävarmempia siirtyvät priorisoinnissa myöhempään vaiheeseen, ja kun niiden toteuttaminen tulee sprinttien edistymisen myötä ajankohtaiseksi, voidaan niiden osalta järjestää tarkemmat määrittelytapaamiset, ja toteutuksen suunnittelu aloittaa.

Tutkimuksessa havaittiin koko aineiston keräämiseen käytettyä aikaväliä tarkastelemalla, että ketterien menetelmien käyttöönottaminen tehosti, selkeytti ja paransi hankkeen tietojärjestelmien toteuttamista merkittävästi. Tämä korostuu järjestelmätoimittajien työskentelyn näkökulmasta tarkasteltuna. Havaintojen perusteella menetelmät tarjosivat projektinhallinnallisesta ja tietojärjestelmien kehittämisen näkökulmasta sellaiset keinot, joilla tutkimuskohteena olevan tietojärjestelmän uudistushankkeen laajuinen kokonaisuus voidaan vaiheistaa ja toteuttaa tehokkaalla toimintatavalla.

4.2 Tulosten luotettavuus ja käyttöarvo

Tutkimuksessa saavutettuja tuloksia voidaan pitää melko luotettavina. Luotettavuutta pystytään tutkimuksen perusteella arvioimaan etenkin järjestelmätoimittajan näkökulmasta. Diplomityön kirjoittaja on toiminut hankkeen ensimmäisestä päivästä alkaen erittäin merkittävässä roolissa olevan järjestelmätoimittajan projektipäällikkönä. Kirjoittaja on työskennellyt hankkeen parissa tutkimustulosten kirjoitushetkellä yhteensä 2950 tuntia, joiden aikana osallistuttuja projektitapaamisia on ollut yli 200 kappaletta. Tutkimusaineiston kerääminen ja analysointi on toteutettu erittäin merkittävältä osin niin lähellä tutkimuskohdetta kuin on ollut mahdollista. Osallistuminen ketterien menetelmien käyttämiseen, niiden käyttöönottamiseen ja menetelmien toimintatapojen hyödyntämiseen on ollut erittäin aktiivista.

Tutkimushavaintojen ja -tulosten voidaan uskoa olevan melko lähellä myös muiden järjestelmätoimittajien näkemyksiä. Tämä johtuu siitä, että tutkimuksen näkökulmasta kirjoittajan edustaman osajärjestelmän kehitystyö ja projektinhallinta on tehostunut ketterien menetelmien myötä merkittävästi. Tällä voidaan uskoa olleen vastaavia vaikutuksia myös muiden osajärjestelmien kehittämiseen. Tämä olettamus perustuu osaltaan siihen, millaisia havaintoja menetelmien käyttämisestä on kirjoittajan toimesta tehty Scrumin mukaisten projektitapaamisten ja viikoittaisten tilannepalaverien yhteydessä.

Asiakkaan näkökulmasta varmuutta ketterien menetelmien soveltuvuudesta hankkeeseen on vaikeampi arvioida ilman täydentävää tutkimusta. Jatkotutkimuksena tässä esitettyjen tutkimustulosten vahvistamiseen ja luotettavuuden arviointiin olisi hyödyllistä toteuttaa kysely- tai haastattelututkimus hankkeen muille osapuolille. Tällä pystytään selvittämään muiden osapuolien näkemys menetelmien käytöstä, sekä niiden tuomista eduista ja mahdollisista haitoista hankkeen edistymiseen. Tuloksia voitaisiin verrata tässä tutkimuksissa saavutettuihin, ja arvioida tulosten luotettavuuden paikkaansa pitävyyttä.

Tutkimustulosten uskotaan olevan melko merkittäviä, sillä ketterien menetelmien käytöstä osana suuria IT-hankkeita on saatavilla hyvin vähän empiriaan perustuvaa tutkimustietoa. Tulosten mukaan menetelmät soveltuvat hyvin usean osapuolen väliseen projektiin, ja ne tarjoavat hyviä periaatteita ja toimintatapoja isojenkin organisaatioiden toimintaan. Merkittävänä jatkotutkimuskohteena hankkeen osalta olisi tutustua tarkem-

min LeSS-periaatteiden mukaiseen, skaalautuvaan Scrum-menetelmän käyttöön. Tämä jatkotutkimus tarjoaisi hankkeeseen mahdollisesti uusia näkemyksiä yksittäisten toteutusvaiheiden projektinhallintaa merkittävämmässä suhteessa, koko projektiorganisaation toiminnasta asiakkaan sisäisistä toimintatavoista alkaen, ottaen huomioon hankkeen jokaisen toteutusvaiheen, tietojärjestelmätason, kansainvälistymisen ja näiden välisen kokonaishankkeen projektinhallinnan. Tämän jatkotutkimuksen yhteydessä olisi hyvä verrata hankkeesta saavutettuja havaintoja LeSS-periaatteen kehittäneiden Larman ja Vodden (2014) julkaisemiin tapaustutkimuksiin hankkeista, joissa LeSS-periaatetta on käytetty osana suurien organisaatioiden toimintaa.

5. YHTEENVETO

Ketterien ohjelmistokehitysmenetelmien hyödyntämisestä isoissa projekteissa on saatavilla varsin vähän tutkimustietoa. Ketterät menetelmät yhdistetään yleensä pieniin ohjelmistokehitysprojekteihin, joissa toteutustyö tehdään yhden tiimin toimesta. Tässä diplomityössä tutkittiin ketterien menetelmien käyttöä osana laajamittaista tietojärjestelmän uudistushanketta. Hankkeeseen osallistuu useita järjestelmätoimittajia, joilla jokaisella on omat vastuualueensa erillisten osajärjestelmien toteuttamisessa, mutta joiden välillä vaaditaan tiivistä yhteistyötä tietojärjestelmäkokonaisuuden toteuttamisen kannalta.

Diplomityössä suoritettiin kattava tapaustutkimus ketterien menetelmien käyttötavoista, ja niiden mukaisten periaatteiden noudattamisesta osana tietojärjestelmähanketta. Tutkimus toteutettiin osana kirjoittajan työskentelyä yhden järjestelmätoimittajan projektipäällikkönä, mikä antoi tutkimushavaintojen tekemiselle hyvän mahdollisuuden, tiiviin projektityöskentelyn ja jatkuvan osallistumisen ansiosta. Tutkimuksessa saavutettiin selkeät vastaukset tutkimusongelmaan, ja pystyttiin vastaamaan työssä esitettyihin kysymyksiin käytetyistä menetelmistä ja niiden käyttötavoista.

Saavutettujen tutkimustulosten perusteella voidaan sanoa ketterien ohjelmistokehitysmenetelmien soveltuvan erinomaisen hyvin osaksi suuria tietojärjestelmähankkeita. Merkittävin havainto tuloksissa on se, että riippumatta projektin koosta, ketterien menetelmien mukaiset periaatteet ja käytännöt soveltuvat sellaisenaan, sekä yksittäisten kehitysprojektien hallintaan että usean osapuolen väliseen laajamittaiseen hankekokonaisuuteen. Menetelmät mahdollistavat projektinhallinnan riippumatta projektin kokoluokasta. Tutkimustulokset antavat hyvän pohjan tutkimuksen laajentamiselle. Havaintojen perusteella jatkotutkimusaiheeksi ehdotetaan tarkempaa ja syvällistä tutustumista *"Large-Scale Scrum"* -periaatteen mukaiseen Scrumin skaalautuvuuteen suuriin projektikokonaisuuksiin.

Tuloksista voidaan johtaa useita konkreettisia suosituksia tutkimuksen kohteena olevan tietojärjestelmähankkeen kehittämiseen. Suosituksilla uskotaan olevan huomattavia vaikutuksia hankkeen jatkon kannalta; tarjoten keinoja osapuolten välisen toiminnan tehostamiseen, projektin seurattavuuden parantamiseen, sprint-suunnitelmien helpompaan tekemiseen sekä onnistuneiden järjestelmäjulkaisuiden saavuttamiseen.

- Tärkein suositus liittyy ketterien menetelmien empiirisyyteen, ja siihen, että menetelmiä on tarkoitus kokeilla ja löytää jokaiseen tilanteeseen parhaiten sopivat toimintatavat. Järjestelmätoimittajien näkökulmasta hanke etenee tällä hetkellä

järkevällä tavalla. Toteutuksessa keskitytään olennaiseen ja se on vaiheistettua. Menetelmien perusperiaatteet siis toimivat, mutta ne voisivat toimia edelleen huomattavasti tehokkaammin ja paremmin.

- Ketterien menetelmien periaatteet ja toimintatavat olisi syytä käydä jokaisen osapuolen kanssa yhteisesti läpi. Kartoittamalla jokaisen toimittajan ja asiakkaan mielipiteet nykyisten toimintatapojen hyvistä ja huonoista puolista, sekä arvioimalla niitä kriittisesti, voidaan hankkeen toimintatapoja lähteä kehittämään. Lean-filosofian, Scrumin ja Kanbanin periaatteiden mukaisesti on tärkeää keskittyä olennaiseen tekemiseen ja kokeilla rohkeasti uusia toimintatapoja ja ideoita, joilla organisaation toimintaa voidaan tehostaa. Sopimalla ja keskustelemalla käytäntöihin liittyvistä asioista voidaan hankkeessa saavuttaa todella merkittäviä lopputuloksia.
- *"Large-Scale Scrum"* -viitekehyksen mukaisiin toimintatapoihin on syytä tutustua. Tämän viitekehyksen esittämiä periaatteita kannattaisi miettiä ihan asiakkaan hankejohdon tasolla, kuin myös järjestelmätoimittajien ja asiakkaan välisen toteutustyön kehittämisessä.

Tässä diplomityössä esitetty tutkimus onnistui kokonaisuutena hyvin. Tutkimuksen suorittaminen ja havaintojen tekeminen oli pitkäaikainen prosessi, mikä mahdollisti tutkimuksen kattavuuden. Työssä saavutettiin selkeät tuloksesta, joista pystyttiin nostamaan selkeät jatkotutkimusaiheet sekä konkreettiset toimenpiteet hankkeen toimintatapojen kehittämiseen.

LÄHTEET

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J. 2002. Agile software development methods. Review and analysis. Espoo. VTT Publications 478. 107 p.

af Hällström, K. 2011. Agile BI - Liiketoimintatiedot hallintaympäristön ketterä rakentaminen. Diplomityö. Tampere. Tampereen teknillinen yliopisto, Tietojohtamisen koulutusohjelma. 64 s. + liitt. 2 s.

Anderson, D. 2010. Kanban - Successful Evolutionary Change for Your Technology Business. Blue Hole Press. 278 p.

Asiakasyrityksen esitys hankkeen välistatustapaamisessa [PPT]. 3.10.2014. Julkaisematon powerpoint-esitys. 11 s.

Beck, K., Beedle, M. Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D. Manifesto for Agile Software Development [WWW]. 2001. [viitattu 3.10.2015]. Saatavissa: http://agilemanifesto.org/

Boehm B. 2002. Get Ready for Agile Methods, with Care. Computer, 35, pp. 64-69.

Burn down chart [WWW]. Wikipedia. 26.6.2009. [viitattu 12.10.2015]. Saatavissa: https://en.wikipedia.org/wiki/Burn_down_chart

Confluence - Team Collaboration Software [WWW]. Atlassian. [viitattu 19.10.2015]. Saatavissa: https://www.atlassian.com/software/confluence/

Enemmän, mutta parempaa globalisaatiota [WWW]. Akavan globalisaatiolinjaukset. Helsinki, Akava. 15.6.2010. [viitattu 10.11.2015]. Saatavissa: http://issuu.com/akava/docs/enemman_mutta_parempaa_globalisaatiota

Flowdock: Group chat for teams [WWW]. Rally Software Development Corp. [viitattu 19.10.2015]. Saatavissa: https://www.flowdock.com/

Haikala, I. & Märijärvi, J. 2006. Ohjelmistotuotanto. 11. painos. Helsinki, Talentum. 440 s.

Hakansuu, P. 2011. Toiminnan muuttaminen strategisen ohjausjärjestelmän avulla. Case: tuloskortin kehittäminen. Maisterin tutkinnon tutkielma. Helsinki. Aalto-yliopisto, Kauppakorkeakoulu, Laskentatoimen ja rahoituksen laitos. 84 s. + liitt. 2 s.

Highsmith, J. 2002. Agile Software Development Ecosystems. Boston, Addison-Wesley. 404 p.

Hirsjärvi, S., Remes, P. & Sajavaara, P. 2007. Tutki ja kirjoita. 13.-14., osin uudistettu painos. Helsinki, Kustannusosakeyhtiö Tammi. 448 s.

Hirvonen, H. 2012. Lean management - prosessijohtamismalli. Hyödyntäminen finanssialan yrityksessä. Opinnäytetyö. Vantaa. Metropolia Ammattikorkeakoulu, Liiketalouden koulutusohjelma. 54 s. + liitt. 2 s.

Huttunen, J. 2006. Ketterän ohjelmistokehitysmenetelmän määrittely, vertailu ja käyttäjäkysely. Diplomityö. Espoo. Teknillinen korkeakoulu, Sähkö- ja tietoliikennetekniikan osasto. 84 s. + liitt. 7 s.

James, M. Scrum Reference Card. 2012. Version 0.9. 6 p. Saatavissa: http://scrumreferencecard.com/

JIRA Software - Issue & Project Tracking for Software Teams [WWW]. Atlassian. [viitattu 19.10.2015]. Saatavissa: https://www.atlassian.com/software/jira

Kalermo, J. & Rissanen, J. 2002. Agile software development in theory and practice. Master's thesis. Jyväskylä. University of Jyväskylä, Department of Computer Science and Information Systems. 188 p.

Kniberg, H. & Skarin, M. 2010. Kanban and Scrum - making the most of both. C4Media. 120 p. Saatavissa: http://www.infoq.com/minibooks/kanban-scrum-minibook

Koski, T. 2012. Kanban: Periaatteita ja kokemuksia. Kandidaatin tutkielma. Jyväskylä. Jyväskylän yliopisto, Tietojenkäsittelytieteiden laitos. 49 s.

Kosonen, S. 2005. Ohjelmoinnin opetus Extreme Programming -hengessä. Pro gradu -tutkielma. Jyväskylä. Jyväskylän yliopisto, Tietotekniikan laitos. 76 s.

Larman, C. & Vodde, B. 2008. Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum. 1st Edition. Addison-Wesley Professional. 368 p.

Larman, C. & Vodde, B. 2010. Practices for Scaling Lean & Agile Development: Large, Multisite, and Offshore Product Development with Large-Scale Scrum. 1st Edition. Addison-Wesley Professional. 624 p.

Larman, C. & Vodde, B. Scaling Scrum, Scaled Agile - Large Scale Scrum (LeSS) [WWW]. The LeSS Company B.V. 2014. [viitattu 20.11.2015]. Saatavissa: https://less.works/

Larman, C. & Vodde, B. LeSS Rules (October 2015) [WWW]. The LeSS Company B.V. 2015. [viitattu 20.11.2015]. Saatavissa: https://less.works/less/rules/index.html

Lehtonen, T., Tuomivaara, S., Rantala, V., Känsälä, M., Mäkilä, T., Jokela, T., Könnölä, K., Kaisti, M., Suomi, S., Isomäki, M. & Ylitoiva, M. Sulautettujen järjestelmien ketterä käsikirja. Turku 2014, TEKES, Turun Yliopisto, Työterveyslaitos. 98 s. Saatavissa: http://trc.utu.fi/embedded/kasikirja

Lekman, L. Mikä ihmeen Kanban? [WWW]. 26.9.2009. [viitattu 13.10.2015]. Saatavissa: http://lekman.fi/2009/09/26/mika-ihmeen-kanban/

Liedenpohja, H. 2006. Ketterät ohjelmistoprosessit ja laadunhallinta. Seminaarityö (luonnos). Helsingin yliopisto, Tietojenkäsittelytieteen laitos.

Lindberg, H. 2003. Extreme Programming. Pro gradu -tutkielma. Tampere. Tampereen yliopisto, Tietojenkäsittelytieteiden laitos. 67 s.

Poimala, S. Ketteryys haltuun: Ketterän kehityksen yleiset periaatteet [WWW]. 6.6.2013. [viitattu 20.11.2015]. Saatavissa: http://www.meteoriitti.com/2013/06/06/ketteryys-haltuun-ketteran-kehityksen-yleiset-periaatteet/

Poppendieck, M. & Poppendieck, T. 2006. Implementing Lean Software Development: From Concept to Cash. Addison-Wesley Professional. 304 p.

Scapens, R. W. 1990. Researching management accounting practice: The role of case study methods. British Accounting Review, 22, pp. 259-281.

Schramm, W. 1971. Notes on Case Studies of Instructional Media Projects. Washington DC. Standford University, Institute for Communication Research. 43 p. Saatavissa: http://files.eric.ed.gov/fulltext/ED092145.pdf

Schwaber, K. 2004. Agile Project Management with Scrum. Microsoft Press. 192 p.

Schwaber, K. & Beedle, M. 2002. Agile Software Development with Scrum. Prentice Hall. 158 p.

Schwaber, K., Leffingwell, D. & Smits, H. 2005. A Playbook for Adopting the Scrum Method of Achieving Software Agility. Scrum Alliance. 53 p.

SharePoint - Team Collaboration Software Tools [WWW]. Microsoft. [viitattu 19.10.2015]. Saatavissa: https://products.office.com/en-us/sharepoint/collaboration

Todman, C. 2001. Designing a Data Warehouse: Supporting Customer Relationship Management. Upper Saddle River, New Jersey, Prentice Hall. 323 p.

Trello [WWW]. Trello, Inc. [viitattu 19.10.2015]. Saatavissa: https://trello.com/

Vuorinen, J. 2011. Scrum-menetelmän käyttö Pirkanmaalaisissa ohjelmistoyrityksissä. Diplomityö. Tampere. Tampereen teknillinen yliopisto, Tietotekniikan koulutusohjelma. 50 s. + liitt. 7 s.

Wester, J. What is Kanban? [WWW]. 2012. [viitattu 13.10.2015]. Saatavissa: http://www.everydaykanban.com/what-is-kanban/

Wirgentius, M. 2003. Agile-menetelmät ja Crystal Clear ohjelmistotuotantoprojektissa. Insinöörityö. Espoo. Espoo-Vantaan teknillinen ammattikorkeakoulu.

Womack, J., Jones, P. & Roos, D. 1991. The Machine That Changed the World: The Story of Lean Production. HarperCollins. 323 p.

Yin, R. 2003. Case Study Research: Design and Methods. 3rd edition. SAGE Publications. 181 p.